MULTIPLE-HYPOTHESIS VISION-BASED LANDING AUTONOMY

A Dissertation Proposal
Presented to
The Academic Faculty

By

Takuma Nakamura

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology

December 2018

MULTIPLE-HYPOTHESIS VISION-BASED LANDING AUTONOMY

Approved by:

Dr. Eric N. Johnson, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Panagiotis Tsiotras
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Eric Marie Feron
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Patricio Antonio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. James Hays
School of Computer Science
Georgia Institute of Technology

Date Approved: July 24, 2018

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one– and preferably only one –obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea – let's do more of those!

Tim Peters

Hello, world!

waves

ACKNOWLEDGEMENTS

I would like to thank my parents, friends, my advisor Dr. Eric Johnson, and the members of his UAVRF, namely Steve Haviland, Dmitry Bershadsky, Lee Whitcher, Kyuman Lee, Yong Eun Yoon, Daniel Magree, Ep Pravitra, and many other lab members for all the support. I would also like to thank Dr. Panagiotis Tsiotras, Dr. Eric Feron, Dr. Patricio Antonio Vela, and Dr. James Hays for their effort and time as my thesis committee members. Lastly, I would like to thank my dearest wife Mariko for all her love and support and thank my upcoming baby Sola for being my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Nomenclature

A	Continuous-time state matrix
F	Discrete-time state matrix
G	Discrete-time input matrix
H	Discrete-time output matrix
x	State
u	Input
y	Output
z	Measurement
ρ	Axis-angle
Φ	Euler-angle
p	Position
v	Velocity
a	Acceleration
ω	Angular velocity
b	Bias
e	Estimation error
s	Specific acceleration
K	Kalman gain matrix
P	Estimated error covariance matrix
Q	Process noise covariance matrix
I	Identity matrix
Σ	Measurement noise covariance matrix
Z	The Z value, Mahalanobis distance
ϵ	Innovation, or measurement residual
σ	Standard deviation of random variable
X, Y	Random variable (uppercase)
$N(\mu, \sigma^2)$	Univariate normal distribution
ν	Zero-mean Gaussian-distributed random variable with the variance of one $\nu \sim N(0, 1)$

Superscripts, Subscripts

b	Body-fixed frame
n	North-east-down (NED) frame
gyro	Gyroscope frame
acc	Accelerometer frame
imu	Inertial measurement unit (IMU) frame
gps	Global positioning system, GPS
mag	Magnetometer
baro	Barometric pressure sensor
$-$	*A priori* variable in the extended Kalman and particle filters
$+$	*A posteriori* variable in the extended Kalman and particle filters
$\hat{a}$	Estimated value of a
$\tilde{a}$	Measured value of a
a^d_{bc}	Vector a from point b to c expressed in d frame
R^b_a	Transformation matrix from a to b coordinate frames

ekf	Parameters used for extended Kalman filters
pf	Parameters used for particle filters
rbpf	Parameters used for Rao-Blackwellized particle filters

Operators

$[a]_\times$	Skew matrix operator. $[a]_\times^T = -[a]_\times$. $a \times b = [a]_\times b$
$\oplus$	Mean update operator of Kalman filters.
$\mathrm{tr}(A)$	Trace of a matrix A.
$\det(A)$	Determinant of a matrix A.

SUMMARY

Unmanned aerial vehicles (UAVs) need humans in the mission loop for many tasks, and landing is one of the tasks that typically involves a human pilot. This is because of the complexity of a maneuver itself and flight-critical factors such as recognition of a landing zone, collision avoidance, assessment of landing sites, and decision to abort the maneuver. Another critical aspect to be considered is the reliance of UAVs on GPS (global positioning system). A GPS system is not a reliable solution for landing in some scenarios (e.g. delivering a package in an urban city, and a surveillance UAV repatriating a home ship with the jammed signals), and a landing solely based on a GPS extremely decreases the UAV operation envelope. Vision is promising to achieve fully autonomous landing because it is a rich-sensing, light, affordable device that functions without any external resource.

Although vision is a powerful tool for autonomous landing, the use of vision for state estimation requires extensive consideration. Firstly, vision-based landing faces a problem of occlusion. The target detected at a high altitude would be lost at certain altitudes while a vehicle descends; however, a small visual target can not be recognized at high altitude. Second, standard filtering methods such as extended Kalman filter (EKF) face difficulty due to the complex dynamics of the measurement error created due to the discrete pixel space, conversion from the pixel to physical units, the complex camera model, and complexity of detection algorithms. The vision sensor produces an unfixed number of measurements with each image, and the measurements may include false positives. Plus, the estimation system is excessively tasked in realistic conditions. The landing site would be moving, tilted, or close to an obstacle. The available landing location may not be limited to one. In addition to assessing these statuses, understanding the confidence of the estimations is also the tasks of the vision, and the decisions to initiate, continue, and abort the mission are made based on the estimated states and confidence. The system that handles those issues

and consistently produces the navigation solution while a vehicle lands eliminates one of the limitations of the autonomous UAV operation.

This thesis presents a novel state estimation system for UAV landing. In this system, vision data is used to both estimate the state of the vehicle and map the state of the landing target (position, velocity, and attitude) within the framework of simultaneous localization and mapping (SLAM). Using the SLAM framework, the system becomes resilient to a loss of GPS and other sensor failures. A novel vision algorithm that detects a portion of the marker is developed, and the stochastic properties of the algorithm are studied. This algorithm extends the detectable range of the vision system for any known marker. However, this vision algorithm produces highly nonlinear, non-Gaussian, and multi-modal error distribution, and a naive implementation of filters would not accurately estimate the states. A vision-aided navigation algorithm is derived within extended Kalman particle filter (PF-EKF) and Rao-Blackwellized particle filter (RBPF) frameworks in addition to a standard EKF framework. These multi-hypothesis approaches not only deal well with a highly nonlinear and non-Gaussian distribution of the measurement errors of vision but also result in numerically stable filters. The computational costs are reduced compared to a naive implementation of particle filter, and these algorithms run in real time. This system is validated through numerical simulation, image-in-the-loop simulation, and flight tests.

CHAPTER 1
INTRODUCTION AND MOTIVATION

This thesis presents a novel architecture for vision-based landing for the use of vertical take-off and landing (VTOL) unmanned aerial vehicles (UAVs). Three filtering techniques (extended Kalman filter, extended Kalman particle filter, and Rao-Blackwellized particle filter) are utilized in a paradigm known as simultaneous localization and mapping (SLAM). Vision as well as other sensors like an inertial measurement unit (IMU), a global positioning system (GPS), a magnetometer, and a barometric pressure sensor are fused in the SLAM paradigm to estimate the state of the vehicle and a landing site. Vision is a sensor that only produces relative position measurements, but being integrated with other sensors, it is capable to map the location of the targets in a navigation coordinate frame, and the vehicle can be localized by capturing the connection to it. This allows the system to produce a navigation solution for landing even while GPS signals are lost or a camera temporarily loses track of a target. This approach is especially powerful for the case of a moving target, in which the system needs to continuously update the landing point until touchdown. However, when a vehicle approaches to a landing location, an entire marker can not be captured by an on-board camera. The altitude when this happens is determined by the size of the marker and the field of view of the camera, but the altitude is often not low enough to make an open-loop control just by predicting the marker location with no measurement. A new vision algorithm is developed to expand the range where a vision system can observe markers. This algorithm helps the SLAM paradigm estimate the states of landing sites from a high altitude to a low altitude even if a part of the markers are occluded or out of the image. This chapter introduces the motivation of this and other related work. The contribution of present research is summarized in the following section, and this chapter is concluded with a guide to the document.

1.1 Motivation

Unmanned aerial vehicles (UAVs) have recently received massive attention because of their unique capability to move freely in three-dimensional space. Adding to this, their affordability and the fact that they do not put the life of a pilot in danger made us replace many of the existing methods with UAVs in various areas from entertainment to military operations. Once their autonomy and regulation become less limiting, more tasks will be operated by UAVs. For example, UAVs could deliver a package to the backyard of a house in an urban city as a fast, cost-effective, and environmentally-friendly alternative to terrestrial transportation. Industries display an interest in parcel delivery, including Amazon.com, DHL [1], and Alphabet's Project Wing. Fire engines could launch UAVs to let them collect data from disaster zones and relay the information back to rescuers. An on-demand autonomous flying taxi could pick you up anywhere and anytime.

Improving the performance of landing is one of the tasks to be completed to realize these applications. Landing is one of the most dangerous operations for any aircraft; in fact, 36 percent of fatal accidents in modern aviation history happened in the phases of final approach and landing [2]. Mismaneuvers and misjudgments are directly connected to a crash. Available space for landing is limited, and a vehicle needs to be precisely controlled. There could be an obstacle on an expected landing site, so a landing trajectory could be modified, or a go-around decision would be required.

One important aspect for a successful landing is the capability to assess the states of the vehicle and a landing site. Highly precise control of a vehicle is implausible without understanding the current state of the vehicle. An appropriate decision can not be made without knowing the condition of the landing site. However, development of a system that accurately and consistently estimates the states of the vehicle and the landing site is a complicated task. The first issue is the reliance of UAV navigation on GPS systems. The GPS system is typically only a reference for an absolute position of a vehicle, and the

position of a vehicle is controlled based on the desired position calculated using the GPS signal. The accuracy of GPS receivers may not be sufficient for precision landing. Multiple paths and jamming would collapse the GPS signals. When a landing target is moving, a GPS-based landing is almost impossible without instrumenting a landing site [3]. In order to provide the necessary precision and robust observations through all phases of a landing, another device is required.

Vision is a promising area to achieve autonomous landing. It is a light, inexpensive, data-rich measurement source. It does not depend on any external signal and unsusceptible to jamming and spoofing. Also, since a human pilot recognizes a landing location using vision, our existing systems are designed to be easily recognized using vision. The capability for a UAV to conduct a vision-based landing would enable the design of a landing facility both manned and unmanned aircrafts could utilize.

Marker observation systems are one of the most important aspects for vision-based landing. Many previous vision-based landing work use a marker, but typically assume two conditions: (1) a visual marker is observable throughout all the landing phases or a time from the loss of the observation to a touchdown is negligibly short, and (2) any custom-prepared marker is available for vision-based landing. UAVs used for vision-based landing are typically equipped with a downward facing camera, and the mission is to land on the ground that has a marker. This assumption may not be valid in realistic conditions. Helipads are typically designed to have the dimensions which are 1.5 times the overall length of an aircraft. Such a large helipad can be recognized at a high altitude; however, it is impossible to see an entire landing location while a vehicle descends (Figure 1.1). A human pilot observes a part of the helipad/runway and understands the entire geometry of the helipad to localize himself/herself. When a UAV conducts a landing using vision, the same capability to localize itself with respect to a landing site is required. One easy solution for this is to place a distinct visual target on a known location. Many researchers use this approach for vision-based landing, but instrumenting a landing site would be a

time-consuming, expensive, and sometimes impossible option. When an algorithm is able to localize the position of the vehicle with respect to a marker from a high altitude to touchdown without a special marker, it is an ideal solution.

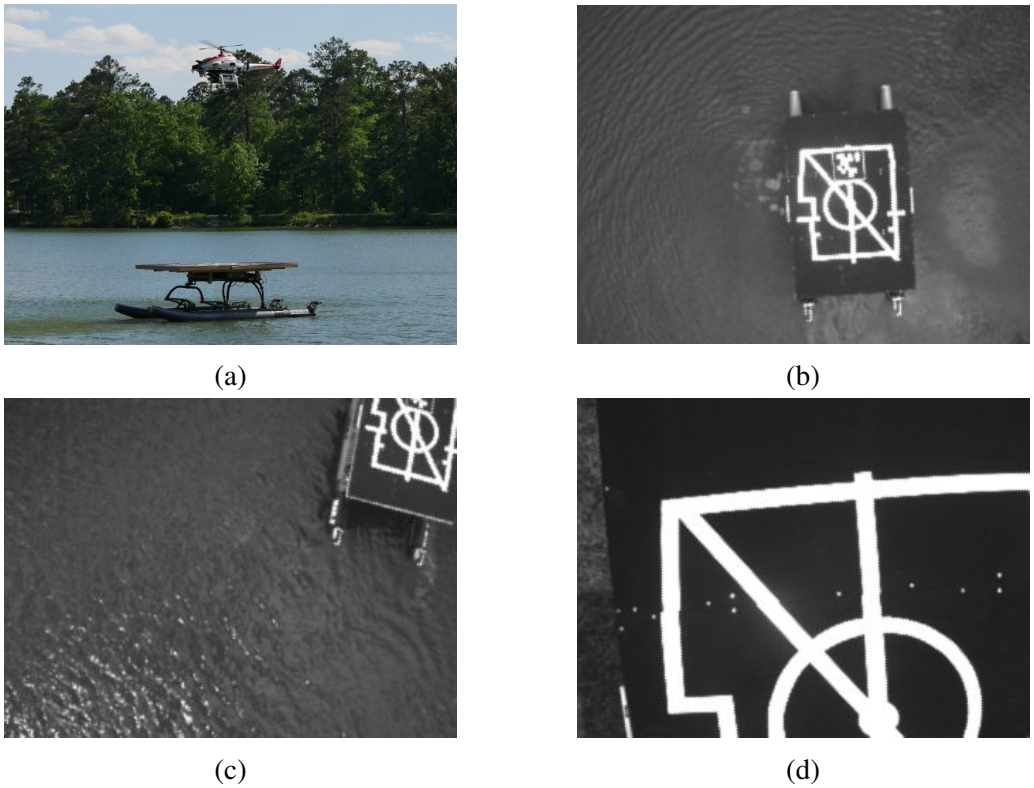
Another key part of the vision-based landing is state estimation architecture. With or without a special marker, a raw detection is not used as a target location, but an estimation algorithm is used to produce a navigation solution. An extended Kalman filter is the most widely used algorithm; however, it assumes a Gaussian-distributed measurement noise, and the dynamics and observations are represented appropriately in a linear system with a linearization. For visual measurements, this is not the case. Measurement noises are highly nonlinear and non-Gaussian. The focus of this work is to develop a system that solves the above problems; that is, a system that

1. can observe a marker from high altitude to low altitude,
2. does not require a custom-prepared marker,
3. estimates the states of the vehicle in a circumstance of GPS failure, and
4. that deals well with inherently complex error distribution of sensors.

1.2 Related works

1.2.1 Landmarks for vision-based landing

Landmarks for autonomous vision-based landing share certain characteristics in common. They are distinct from the background scenery, contain enough features for a system to extract the pose of a marker, and can be detected from a high altitude to a low altitude. Merz, et al.[4] designed a landmark consisting of multiple groups of circles. Lange, et al.[5] and Verbandt, et al.[6] created a landmark consisting of multiple concentric white rings, and Jung, et al.[7] utilized a landmark that consists of three pairs of circular targets on an H-shaped helipad. The landmarks used in these works are shown in Figure 1.2. There



(a) (b)

(c) (d)

Figure 1.1: Figure (a) shows a Yamaha RMAX helicopter chasing a boat using its vision system, and Figure (b), (c), and (d) show an on-board image obtained in a flight test. In order to track the target until touchdown, a small AprilTag is placed on an expected landing location. In this kind of vision-based mission, occlusions happen in multiple ways. For example, the camera could lose the entire target from its line of sight when chasing (as shown in Figure (c)), and another scenario is that the target does not fit in the on-board image as a vehicle descends as shown in Figure (d).

are other types of the more complex landmarks for the application of autonomous landing such as [8] that consists of different rectangle sizes. Although these landmarks have a variation in sizes and shapes (e.g. circles ([4], [5], [6], [9]), rectangles ([8], [10]), H-shape ([11], [12], [13], [14]) T-shape ([15]), etc), there is a common idea behind the design of the landmarks for vision-based landing. That is to try to use the existing landmarks for manned aircraft. When a custom-prepared marker is available, a marker is designed to be detected at various altitudes from the initiation of the landing to the touchdown. Many of the recent research use AprilTags [16] for vision-based landing ([17], [18], [19]). Serra, et al. [20]

used other types of fiducial markers ARToolkit library[1] provides. The author's previous works also utilized an H-shape helipad ([21], [22], [23]), circle [24], and AprilTag [25].

(a) (b) (c)

Figure 1.2: Figure (a), (b), and (c) show a landmark used in [4], [5], and [7], respectively. Landmarks for autonomous vision-based landing is designed so that the vision system can detect a landmark from a high altitude to a low altitude. In order to achieve this, various sizes of the objects are placed on a landmark.

1.2.2 Detection of landing locations

Detection frameworks vary for each landmark. For example, [4] detects contours from a given image and [8] detects corners of objects. These primitive algorithms would function on a limited condition; however, it would not acclimate well to cluttered or featureful backgrounds or poor lighting conditions. Successful state-of-the-art detection frameworks are typically obtained through training in machine/deep-learning architectures. Viola and Jones invented a robust real-time detection framework [26] using Haar-like features [27]. Vision-based landing using the Viola-Jones framework include [28], [21], and [22]. The more recent detectors ([29], [30]) use a support vector machine (SVM) ([31]) to train a detector using the extracted features of the target, and [6] and [32] utilize the SVM for a classifier. Most of these learning-based approaches detect a target with a sliding window approach. This method slides a small box around the given image, classifying each crop inside the box. The size of the box is changed once the operation is iterated over the entire image. Although it has not been used for vision-based landing, some deep-learning

[1] http://www.hitl.washington.edu/artoolkit/

algorithms detect targets without the sliding window ([33], [34], [35]), but objects are detected with a corresponding bounding box. The images are evenly segmented to measure an objectness of each box, and the object intersecting multiple boxes are bounded by an oversized box. Most of the modern machine/deep-learning detection framework face the same issue of discontinuity due to box size, stride size, and misaligned bounding box. There are algorithms that detect an object in pixel-level precision, but these algorithms are typically too slow or need a special computational power to run in real-time ([36], [37]).

1.2.3 Landing on a moving platform

A camera frequently loses track of a moving target due to fast vehicle movement, shadows, poor lighting conditions, or occlusions. In order to compensate for a temporary loss of measurements, it is important to use a model-based estimator ([38]) such as a complementary filter [39], [40], extended Kalman filter (EKF), unscented Kalman filter (UKF) [41], or a particle filter (PF). All three techniques are also popular in SLAM research; examples include [42] and [43], which are EKF-based, and [44], which is PF-based. Some researchers showed that it is possible to land on a marker using vision, but one pitfall for a moving target is the use of optical flow for accurate velocity. Yang, et al. [45] integrated optical flow measurements with IMU and GPS in UKF framework, but this is problematic for a mobile target since it miscomputes the ground speed of a vehicle. This fact disallows many of the popular keyframe-based visual SLAM paradigms, including ORB-SLAM [46] and other SLAM methods ([47], [48], [49], and [50]). This implies that the vision-based landing utilizing these visual SLAM methods (e.g. ORB-SLAM to localize a target [14]) can not be extended to a mobile target. Many of the successful vision-based landings integrate marker measurements in a Kalman filter framework ([10], [17]). There are algorithms that detect and track a target with just vision ([51], [52], [53], [54], [55]); however, for a vision-landing, rather than tracking with a model-free approach, a model-based approach with the fusion of other on-board sensors produces a more accurate solution. Also, landing tar-

gets are typically specific and distinct, and target detection with a single shot is relatively accurate; thus, a tracking-by-detection approach is preferred over a detection-by-tracking approach.

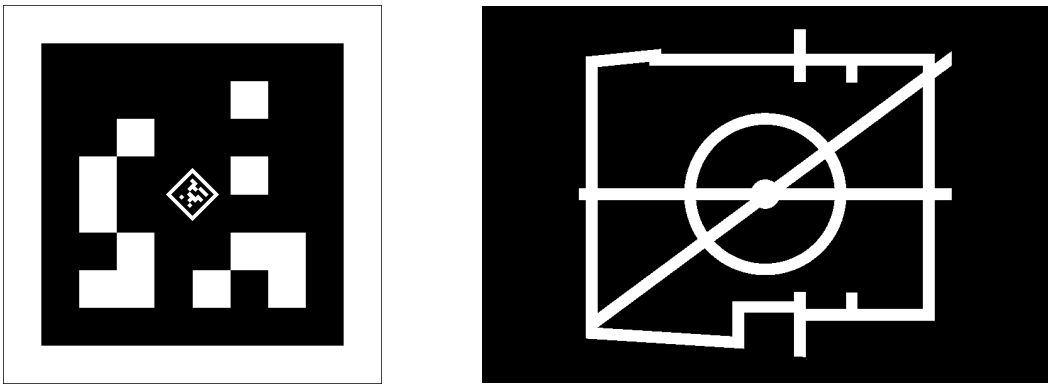
1.2.4 Summary of literature review and comparison with the suggested method

Many researchers work on vision-based autonomous landing, and most of the work in this area focuses on landing on a known target. This is because a system has the visual information of expected landing sites in realistic conditions. The assumption of the controllability of landmarks varies for each work. A custom-made landmark would be installed in some cases, but it is a time-consuming and expensive option. In some scenarios, it is impossible to install a new landmark, and a vision system would be forced to utilize the existing one. Many researchers work on autonomous landing on a standard H-shaped helipad for this reason. In both cases, the common procedure for assessing the landing sites is:

1. Detect a landing site utilizing *a priori* knowledge of a landmark;
2. Estimate the pose, and optionally, the velocity of a landing site using sequential observations;
3. Use Model-based estimation to produce a navigation solution and confidence of estimation.

The architecture proposed in this document follows the same procedure. However, what distinguishes the present work from previous work is threefold: (1) the use of multiple-hypothesis approaches explicitly deals with the nonlinear and non-Gaussian measurement distribution, (2) multiple targets of the 9-DOF (attitude, position, and velocity) states are estimated in the visual SLAM paradigm, and (3) custom-made detection frameworks for vision-based landing that allow a system to observe a marker even when a camera is very close to target. Two types of multiple-hypothesis techniques are implemented: extended

Kalman particle filter (PF-EKF) and Rao-Blackwellized particle filter (RBPF). The detection algorithm of the proposed method considers the same two cases as the previous researchers: when an arbitrary custom-made landmark can be prepared, and when an existing landmark is used. In the present work, these two cases correspond to a nested AprilTag and navy deck pattern as shown in Figure 1.3. The work of this thesis offers the greatest advantage when a custom-prepared marker can not be utilized; however, every vision-aided method includes stochastic properties, to which Gaussian approximation is not appropriate. Thus, the other cases are also able to use this study.



(a) (b)

Figure 1.3: Figure (a) shows a nested AprilTag, and (b) shows a navy deck pattern.

1.3 Thesis contributions

The primary contributions of the thesis are fourfold:

1. Derives vision-aided navigation algorithms within EKF, PF-EKF, and RBPF capable of running in real-time with imagery obtained from a monocular vision camera. These algorithms estimate the states of the vehicle using vision and map the position, velocity, and attitude of the target utilizing the SLAM framework. The effectiveness of the hybrid methods of EKF and PF for SLAM applications are demonstrated in multiple studies [56], [57], including the well-known FastSMAM approach [58]. The originality of the design proposed in this thesis is that the state vector includes

the velocity of the vehicle and targets as well as the sensor biases, which are more appropriate design for fast-dynamics applications such as UAVs. These hybrid methods are reported to be successful particularly in marginalization of vision measurements [44], [59], but to my knowledge, this study is the first to test the marginalization of a full state estimator for both an aircraft and a separate moving platform that includes vision measurements relative to the moving platform. Feasibility of the algorithms is demonstrated with image-in-the-loop simulation and flight test data.

2. Develops a method for detecting a portion of a landmark when a camera is too close to see an entire landmark. The details of the implementation of the method are provided, and the stochastic properties of this method are studied.

3. Proposes a new landing architecture integrating the newly developed detection algorithm in the framework of SLAM. This integration extends the detectable range of the vision system for any known marker and improves the accuracy of the estimation during landing by detecting a target until the last minute of touchdown.

4. Provides numerous test data across three different algorithms (EKF, PF-EKF, and RBPF) to give insights into the advantages and disadvantages of each method. These data display the performance of the algorithms from the perspectives of estimation errors, computational costs, numerical stability, multi-target tracking capability, and robustness to false positives.

1.4 Guide to this document

This paper is organized as follows: Chapter 1 introduces the motivation of this research and related previous work; Chapter 2 reviews preliminary mathematics used to implement visual SLAM algorithm in this work; Chapter 3 derives the vision-aided navigation algorithm within EKF, PF-EKF, and RBPF; Chapter 4 describes the vision algorithms developed to observe a visual target at a very low altitude, in addition to visual measurements to be

fused in navigation algorithms; Chapter 5 discusses the results produced by three types of experiment methods (a numerical simulation, image-in-the-loop simulations, and flight test), the results of which include the analysis on estimation errors, computational cost, numerical stability, the capability to track multiple targets across three different filter designs; and Chapter 6 concludes this document and summarizes the contribution of this research, identifying any promising future directions to pursue.

CHAPTER 2
PRELIMINARIES

This chapter introduces key mathematics and algorithms utilized in this work. Firstly, Lie groups and Lie algebras, focusing on special orthogonal groups SO(3), are explored. These mathematics are utilized to present the attitude of vehicles and landing markers. Secondly, the angular rate is reviewed to explicitly show the used approximations and the relationship to the attitude. In the following section, matrix calculus is reviewed. This section focuses on the derivation of Jacobian matrix, which is used in the propagation of error covariance and the measurement update of the Kalman filter. In the last two sections, extended Kalman filter (EKF) and particle filter (PF) are derived.

2.1 Lie group and Lie algebra

Space rotations can be expressed in multiple ways due to their three degrees of freedom. This thesis uses three representations: Euler angles (Φ), axis-angles (ρ), and direction cosine matrix, (R). Another common representation is unit quaternion, which is also known as Euler-Rodrigues parameters. Chapter 2.2 of [60] presents the summary of each representation. This thesis utilizes axis-angle representations because they are the only form of attitude that enables space rotation to be linearly interpolated [61], which is very preferred for designing the framework of an extended Kalman filter. For example, if there are three frames: α, β, and γ, axis-angle representations can describe the relative attitudes of the three frames as follows:

$$\rho_{\gamma}^{\beta} = k\rho_{\alpha}^{\beta}, \tag{2.1}$$

$$\rho_{\alpha}^{\gamma} = (1-k)\rho_{\alpha}^{\beta}. \tag{2.2}$$

However, axis-angle representations have a singularity at $||\rho|| = 2n\pi$ [62]. Because both propagation and measurement updates are operated by moving through time in small steps, the possible change in rotation on each update is considered to be small. This change occurs inside of the ball of radius of 2π. Therefore, for the work conducted in this thesis, the limitations of axis-angle representations regarding singularities have a negligible impact. The group of rotations in 3-D space is expressed as SO(3) in this document. SO(3) has the following properties:

$$R \in SO(3), \tag{2.3}$$

$$R^{-1} = R^T, \tag{2.4}$$

$$\det(R) = 1. \tag{2.5}$$

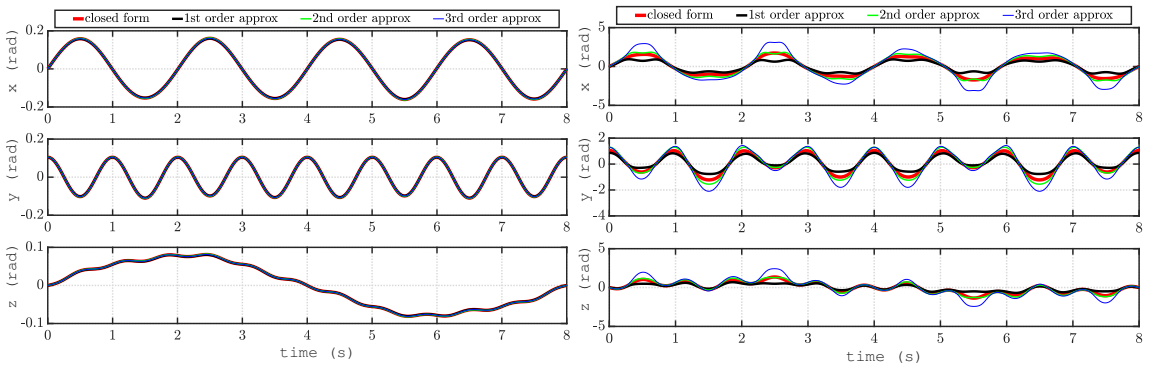
SO(3), the special orthogonal group, is called 'special' because of its positive determinant (2.5). When other isometry transformations such as inverting space are included, the condition of the determinant is loosened to $\det(R) = \pm 1$, and they become a standard orthogonal group O(3), of which SO(3) is a subgroup. Axis-angle representations, $[\rho]_\times \in so(3)$, are the elements of the associated Lie algebra. The exponential map $\exp : so(3) \to SO(3)$ computes a rotation matrix from an axis-angle, and the log map $\log : SO(3) \to so(3)$ computes an axis angle from a rotation matrix. These mappings are defined as follows:

$$R = \exp([\rho]_\times) := \sum_{k=0}^{\infty} \frac{1}{k!}[\rho]_\times^k \tag{2.6}$$

$$[\rho]_\times = \log(R) := -\sum_{k=1}^{\infty} \frac{1}{k}(I - R)^k. \tag{2.7}$$

The definition of the exponential map (2.6) is shown in [63], and the definition of log map (2.7) is shown in [64]. However, they are not a practical way to compute mappings; in

fact, the log map (2.7) can be used only if R is sufficiently close to the identity. Figure 2.1 shows the results of log map using the finite form representation and the power series approximation. When the magnitude of rotation is small, all three approximations (1st-order, 2nd-order, and 3rd-order approximations) successfully convert SO(3) to so(3). However, when the magnitude of rotation is large, these power-series approximations become unstable, and the solutions are not guaranteed to converge even when higher-order approximations are used. The finite closed form formulas are derived in the following subsections. When $||\rho|| \leq \pi$, the exponential mapping can be utilized either locally, which means to represent attitude incrementally between two nearby frames [65], or globally, meaning to represent a total rotation with respect to a reference one [66].



(a) Small angle rotation

(b) Large angle rotation

Figure 2.1: Figure shows the logarithm of rotation matrices computed by the closed-form and power series.

2.1.1 Cross product properties

Cross products and skew-symmetric matrices are closely related; in fact, when using a skew-symmetric matrix, a cross product is written as follows:

$$a \times b = [a]_{\times} b. \tag{2.8}$$

The operator $[a]_\times$, which is denoted $\hat{a}$ (*hat* operator) or $a_\times$ in some literature, maps a vector $a \in \mathbb{R}^3$ to a skew-symmetric matrix $[a]_\times \in so(3)$ as follows:

$$[a]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} = A. \tag{2.9}$$

The inverse of this map, which extracts the components of the vector a from a skew-symmetric matrix $[a]_\times$ is called *vee* operation [67]. The *vee* operation $\vee : so(3) \to \mathbb{R}^3$ is expressed as follows:

$$A^\vee = [a]_\times^\vee = \frac{1}{2} \begin{bmatrix} A(3,2) - A(2,3) \\ A(1,3) - A(3,1) \\ A(2,1) - A(1,2) \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = a. \tag{2.10}$$

As (2.9) and (2.10) indicate, there exists a one-to-one map that preserves the vector space structure through *hat* and *vee* operations (i.e. the vector space $\mathbb{R}^3$ and so(3) are isomorphic). Therefore, $[\rho]_\times \in so(3)$ is often shorthanded as $\rho \in so(3)$.

The rest of this subsection enumerates some of the properties of cross products and skew-symmetric matrices that are used in this thesis without proof.

$$[a]_\times b = -[b]_\times a. \tag{2.11}$$

$$[a]_\times [b]_\times = ba^T - (a \cdot b)I. \tag{2.12}$$

$$(Ga) \times (Gb) = \det(G) G^{-T} (a \times b), \tag{2.13}$$

where $G \in \mathbb{R}^{3\times3}$. When a G is a rotation matrix, this is simplified as follows:

$$(Ra) \times (Rb) = R(a \times b), \tag{2.14}$$

where $R \in SO(3)$.

2.1.2 Exponential Map

The conversion from an axis-angle representation, also known as an axis-magnitude vector [68], to a rotation matrix is computed by the finite form formula called the Rodrigues formula. Remember the familiar Rodrigues formula:

$$R = \exp([\rho]_\times) = I + \left(\frac{\sin\theta}{\theta}\right)[\rho]_\times + \left(\frac{1-\cos\theta}{\theta^2}\right)[\rho]_\times^2, \tag{2.15}$$

where $R \in SO(3)$ and $[\rho]_\times \in so(3)$. This formula is sometimes written (e.g. [68]) as

$$R = \cos\theta I + \frac{\sin(\theta)}{\theta}[\rho]_\times + \frac{1-\cos\theta}{\theta^2}[\rho]_\times[\rho]_\times^T, \tag{2.16}$$

since

$$[\rho]_\times^2 = [\rho]_\times[\rho]^T - \theta^2 I \ (\because \ (2.12)). \tag{2.17}$$

That means the exponential map above yields a rotation by θ radians around the axis given by ρ. The opposite operation, log map, can be derived from the Rodrigues formula:

$$\begin{aligned} R^T &= I + \left(\frac{\sin\theta}{\theta}\right)[\rho]_\times^T + \left(\frac{1-\cos\theta}{\theta^2}\right)[\rho]_\times^{2T} \\ &= I - \left(\frac{\sin\theta}{\theta}\right)[\rho]_\times + \left(\frac{1-\cos\theta}{\theta^2}\right)[\rho]_\times^2 \ (\because [\rho]_\times^T = -[\rho]_\times). \end{aligned} \tag{2.18}$$

Therefore,

$$R - R^T = \frac{2\sin\theta}{\theta}[\rho]_\times. \tag{2.19}$$

That leads to

$$[\rho]_\times = \log(R) = \frac{\theta}{2\sin\theta}(R - R^T). \tag{2.20}$$

This operation transforms $SO(3)$ to $so(3)$. Indeed, $f_{\mathrm{SO3}}(M) = (M - M^T)/2$ is the Euclidean matrix projection onto skew-symmetric matrices [69]. There are many important properties of axis-angle representations, but one important property derived from (2.20) is

$$[\rho_b^n]_\times = -[\rho_n^b]_\times. \tag{2.21}$$

Let $||\rho_b^n|| = \theta_b^n$. By using (2.20),

$$\begin{aligned}
[\rho_b^n]_\times &= \frac{\theta_b^n}{2\sin\theta_b^n}(R_b^n - R_b^{nT}) \\
&= \frac{\theta_b^n}{2\sin\theta_b^n}(R_n^{bT} - R_n^b) \\
&= -\left(\frac{\theta_b^n}{2\sin\theta_b^n}(R_n^b - R_n^{bT})\right) \\
&= -\left(\frac{\theta_n^b}{2\sin\theta_b^n}(R_n^b - R_n^{bT})\right) \ (\because \theta_{nb} = ||\rho_{nb}|| = ||\rho_{bn}|| = \theta_{bn}) \\
&= -[\rho_n^b]_\times.
\end{aligned} \tag{2.22}$$

The inverse of rotation is obtained just by negating axis-angle representations.

2.1.3 Computing a magnitude of rotation

When computing a magnitude of a rotation from an axis-angle representation, the norm of the vector provides the magnitude of the rotation. However, when it comes to computing an axis-angle from a rotation matrix, the magnitude of the rotation is not that simple. The below lemma is used, and in this subsection, this lemma is proved.

$$\mathrm{tr}(R) = 1 + 2\cos(\theta), \tag{2.23}$$

where θ is the magnitude of the rotation given by the rotation matrix R. The proof begins with the Euler-Rodrigues formula:

$$R \cdot x = (\cos(\theta)[x - a(a \cdot x)] + a(a \cdot x) + \sin(\theta)[a \times x]. \tag{2.24}$$

This is true for an arbitrary position vector x. The a is a unit axis about which the rotation matrix R rotates. By differentiating both sides of this Euler-Rodrigues formula, the following equation is obtained:

$$R = \cos(\theta)(I - a^T a) + a^T a + \sin(\theta)[a]_\times. \tag{2.25}$$

Equation (2.25) holds no matter what basis is used. If, however, one sets up a basis such that the 3-direction is coincident with a, then

$$a_1 = a_2 = 0, a_3 = 1. \tag{2.26}$$

This simplifies (2.25) to

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.27}$$

By taking the trace of the both sides of (2.27), the lemma (2.23) is derived. The Euler-Rodrigues formula is interpreted such that any rotation can be expressed as (2.27) by choosing an appropriate rotation basis. More precisely, any rotation matrix R can be transformed to other coordinates as follows:

$$R = T^{-1}R'T = T^{-1} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} T. \tag{2.28}$$

Taking the trace of both sides of (2.28) provides the following:

$$\mathrm{tr}(R) = \mathrm{tr}(T^{-1}R'T) = \mathrm{tr}(R'TT^{-1}) = \mathrm{tr}(R') = 1 + 2\cos(\theta). \tag{2.29}$$

Therefore, the magnitude of a rotation is computed as follows:

$$\theta = \cos^{-1}\left(\frac{\mathrm{tr}(R) - 1}{2}\right). \tag{2.30}$$

More detailed proof is found in [70]. This formula is also given in [67]. Note that θ is just a norm of ρ for exponential map, but θ for log map is bounded in $-\pi \leq \theta \leq \pi$.

2.2 Angular rate

The angular velocity, or angular rate, is denoted w_{nb}^{b} in this document. This is the rate of rotation of the body-fixed frame axes with respect to the local north-east-down (NED) frame axes expressed in the body-fixed frame axes. The time derivative of a rotation matrix can be expressed by using the angular velocity vector. The time derivative of a coordinate transformation matrix is defined ([71], [72]) as follows:

$$\frac{dR_b^n}{dt} = \dot{R}_b^n = \lim_{\delta t \to 0}\left(\frac{R_b^n(t+\delta t) - R_b^n(t)}{\delta t}\right). \tag{2.31}$$

The body-fixed frame, b, is rotating with respect to a stationary reference frame, n. The rotation matrix at time $t + \delta t$ may be expressed as follows:

$$R_n^b(t+\delta t) = R_{b(t)}^{b(t+\delta t)} R_n^b(t). \tag{2.32}$$

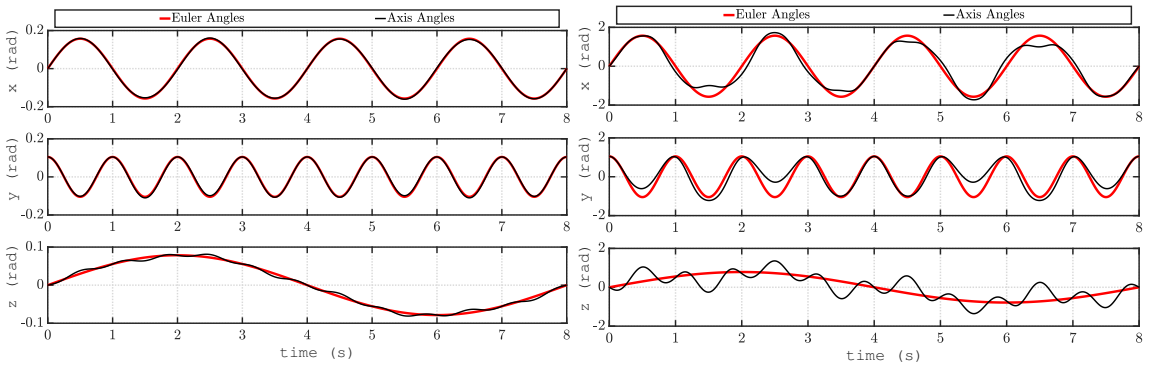
Therefore,

$$R_b^n(t+\delta t) = (R_{b(t)}^{b(t+\delta t)} R_n^b(t))^T = R_b^n(t) R_{b(t+\delta t)}^{b(t)}. \tag{2.33}$$

For an infinitesimally small time δt, the rotation matrix can be approximated as follows [61]:

$$R_{b(t)}^{b(t+\delta t)} \approx I + [\Phi_{b(t)}^{b(t+\delta t)}]_\times \tag{2.34}$$
$$\Rightarrow R_{b(t+\delta t)}^{b(t)} = I - [\Phi_{b(t+\delta t)}^{b(t)}]_\times.$$

where Φ represents the Euler angles. This is because axis-angle representations and Euler angles are identical in the small angle approximation. Figure 2.2 shows the results of



(a) Small angle rotation

(b) Large angle rotation

Figure 2.2: Axis-angles and Euler angles with various magnitudes of rotations.

numerical simulations. The Euler angles are generated by sinusoidal functions, and the exact axis-angles are computed and compared with the Euler angles. When the magnitude of rotation is small, the axis angles are almost the same as Euler angles; however, when the magnitude of rotations is large, this is not the case. Throughout this thesis δt in (2.34) is sufficiently small to use this approximation. Also, the angular velocity vector can be approximated by using this Euler angle vector if δt is infinitesimally small:

$$\Phi_{b(t)}^{b(t+\delta t)} \approx w_{nb}^b \delta t \tag{2.35}$$
$$\Rightarrow \Phi_{b(t+\delta t)}^{b(t)} = -w_{nb}^b \delta t.$$

Finally, (2.31) becomes

$$\begin{aligned}
\dot{R}_b^n &= \lim_{\delta t \to 0} \left(\frac{R_b^n(t+\delta t) - R_b^n(t)}{\delta t} \right) && (2.36)\\
&= \lim_{\delta t \to 0} \left(\frac{R_b^n(t)(I - [\Phi_{b(t+\delta t)}^{b(t)}]_\times) - R_b^n(t)}{\delta t} \right) \ (\because (2.34)) \\
&= \lim_{\delta t \to 0} \left(\frac{R_b^n(t)(I + [w_{nb}^b]_\times \delta t) - R_b^n(t)}{\delta t} \right) \ (\because (2.35)) \\
&= R_b^n(t)[w_{nb}^b]_\times .
\end{aligned}$$

Since $w_{nb}^b = -w_{bn}^b$, the following is also true:

$$\begin{aligned}
\dot{R}_b^n &= R_b^n(t)[w_{nb}^b]_\times && (2.37)\\
&= -R_b^n(t)[w_{bn}^b]_\times . && (2.38)
\end{aligned}$$

The skew-symmetric matrix above is converted from one to another frame as follows:

$$[w_{nb}^n]_\times = R_b^n [w_{nb}^b]_\times R_n^b . \tag{2.39}$$

This derivation largely follows 2.3.1 of [61]. From (2.37),

$$\begin{aligned}
R_b^n(t+\delta t) &= R_b^n(t) + \dot{R}_b^n(t)\delta t && (2.40)\\
&\approx R_b^n(t) + R_b^n(t)[\omega_{nb}^b]_\times \delta t && (2.41)\\
&= R_b^n(I + [\omega_{nb}^b]_\times \delta t). && (2.42)
\end{aligned}$$

From the above observation, $[\omega_{nb}^b]_\times$ is the tangent space at the identity of the rotation group $SO(3)$, and it is transported to any rotation in $SO(3)$ by a multiplication of a rotation matrix.

2.3 Matrix calculus

This section reviews some useful facts of matrix calculus. This is not a comprehensive review of matrix calculus but focuses on the theories useful to explaining the following sections.

2.3.1 Useful lemmas

$$\frac{\partial g^T}{\partial x} = \left(\frac{\partial g}{\partial x}\right)^T, \tag{2.43}$$

$$\frac{\partial g}{\partial x^T} = \left(\frac{\partial g}{\partial x}\right)^T, \tag{2.44}$$

$$\frac{\partial g^T}{\partial x^T} = \frac{\partial g}{\partial x}, \tag{2.45}$$

$$\frac{\partial Ax}{\partial x} = A, \tag{2.46}$$

$$\frac{\partial x^T A}{\partial x} = A. \tag{2.47}$$

$$\frac{\partial \mathrm{Tr}(ABA^T)}{\partial A} = AB^T + AB, \tag{2.48}$$

especially when B is symmetric,

$$\frac{\partial \mathrm{Tr}(ABA^T)}{\partial A} = 2AB. \tag{2.49}$$

The product rule of differential calculus applies to any bilinear operation; therefore, it is also true for the cross product:

$$\frac{\partial}{\partial t}(a \times b) = \left(\frac{\partial a}{\partial t}\right) \times b + a \times \left(\frac{\partial b}{\partial t}\right). \tag{2.50}$$

For any skew matrix $[\rho]_\times$ and $R \in SO(3)$, the following transformation is true:

$$R[\rho]_\times R^T = [R\rho]_\times. \tag{2.51}$$

(2.51) is derived from the property of cross product (2.14). For any arbitrary vectors s and r, the following holds

$$R(s \times r) = (Rs) \times (Rr) \; (\because \; (2.14)), \tag{2.52}$$

where $R \in SO(3)$. Consider vectors satisfying $a \times b = c$:

$$[a]_\times b = c \tag{2.53}$$

$$R[a]_\times b = Rc \tag{2.54}$$

$$R[a]_\times (R^T R) b = Rc \; (\because R^T R = I). \tag{2.55}$$

On the other hand,

$$[a]_\times b = c \tag{2.56}$$

$$R([a]_\times b) = Rc \tag{2.57}$$

$$[Ra]_\times (Rb) = Rc \; (\because (2.52)) \tag{2.58}$$

From (2.55) and (2.58):

$$R[a]_\times R^T Rb = [Ra]_\times Rb \tag{2.59}$$

$$\Rightarrow R[a]_\times R^T = [Ra]_\times \tag{2.60}$$

For an arbitrary vector a and an arbitrary rotation matrix $R \in SO(3)$, (2.51) has been proven. Another important property of a matrix $R \in SO(3)$ is

$$\left(R^T \frac{\partial R}{\partial t}\right)^T = -R^T \frac{\partial R}{\partial t}, \tag{2.61}$$

where t is time. Since $R^T R = I$ for any t,

$$\frac{\partial}{\partial t}(R^T R) = \frac{\partial R^T}{\partial t} R + R^T \frac{\partial R}{\partial t} = 0. \tag{2.62}$$

Therefore,

$$\begin{aligned} \left(R^T \frac{\partial R}{\partial t}\right)^T &= \left(\frac{\partial R}{\partial t}\right)^T \left(R^T\right)^T \\ &= \frac{\partial R^T}{\partial t} R \\ &= -R^T \frac{\partial R}{\partial t} \ (\because (2.61)). \end{aligned} \tag{2.63}$$

2.3.2 Derivative of rotation matrix

As shown in Section 2.2, the local rotation of SO(3) can be replaced by its linearized version, whose vector space is the tangent space of the SO(3). Although there are formulas for the derivative of a rotation at an arbitrary element [63], [73], [74], the work here utilizes the locally approximated version. Particularly, all the derivatives of rotation matrices are expressed in either a body-fixed frame or a marker-fixed frame. When a derivative of rotation around the body frame is computed, differentiation by the rotation parameter is performed

by implicitly multiplying an infinitesimally small rotation around the body frame. When there is an infinitesimal rotation around the body, it is expressed as

$$R_n^{b+} = R_b^{b+} R_n^b = \exp([\rho]_\times) R_n^b. \tag{2.64}$$

When the $\exp([\rho]_\times)$ above is defined as the left-tangent-space perturbation, the counterpart is expressed as follows:

$$R_{b+}^n = R_b^n R_{b+}^b = R_b^n (\exp([\rho]_\times))^T = R_b^n \exp([\rho]_\times^T). \tag{2.65}$$

The left-versus-right perturbation is discussed in 11.1.1 of [75]. Note that the above transformation uses $\exp([\rho]_\times)^T = \exp([\rho]_\times^T)$. This holds for any matrix X. Since $(X^n)^T = (X^T)^n$,

$$\exp(X^T) = \sum_{n=0}^{\infty} \frac{1}{n!} (X^T)^n = \sum_{n=0}^{\infty} \frac{1}{n!} (X^n)^T = \left(\sum_{n=0}^{\infty} \frac{1}{n!} X^n \right)^T = (\exp(X))^T. \tag{2.66}$$

The derivative of a rotation matrix evaluated at a body-fixed frame is defined as follows:

$$\frac{\partial R_n^b}{\partial \rho_n^b} = \left. \frac{\partial \exp([\rho]_\times) R_n^b}{\partial \rho} \right|_{\rho=0}. \tag{2.67}$$

This partial differentiation can be solved using the generators, which is the derivative of rotation around each of the standard axes. Let G_1, G_2, and G_3 denote the derivative around the x, y, and z axes, respectively. Then,

$$G_i = \left. \frac{\partial}{\partial \rho_i} \exp([\rho]_\times) \right|_{\rho=0} = [e_i]_\times, \tag{2.68}$$

$$G_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_\times = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{2.69}$$

$$G_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_\times = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \tag{2.70}$$

$$G_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_\times = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.71}$$

The column of the derivative is presented using these generators. Since the rotation parameter of this work is mostly ρ_b^n and ρ_m^n, which are exponential-mapped as $R_b^n = \exp([\rho_b^n]_\times)$ and $R_m^n = \exp([\rho_m^n]_\times)$, the derivatives evaluated locally around the body frame and the marker frame are useful. For example, for any arbitrary vector $u \in \mathbb{R}^3$, which is not a function of ρ_b^n, differentiating the action of SO(3) on vector space $\mathbb{R}^3$ is computed as follows:

$$\begin{aligned} \frac{\partial \exp([\rho_n^b]_\times)u}{\partial \rho_b^n} &= -\frac{\partial \exp([\rho_n^b]_\times)u}{\partial \rho_n^b} \\ &= -\left.\frac{\partial \exp([\rho]_\times)R_n^b u}{\partial \rho}\right|_{\rho=0} \\ &= -(G_1(R_n^b u)|G_2(R_n^b u)|G_3(R_n^b u)) \\ &= -\left(\begin{bmatrix} 0 \\ -(R_n^b u)_3 \\ (R_n^b u)_2 \end{bmatrix} \middle| \begin{bmatrix} (R_n^b u)_3 \\ 0 \\ -(R_n^b u)_1 \end{bmatrix} \middle| \begin{bmatrix} -(R_n^b u)_2 \\ (R_n^b u)_1 \\ 0 \end{bmatrix}\right) \\ &= [R_n^b u]_\times. \end{aligned} \tag{2.72}$$

Similarly,

$$\begin{aligned}\frac{\partial \exp([\rho_b^n]_\times)u}{\partial \rho_b^n} &= -\left.\frac{\partial R_b^n \exp([\rho]_\times^T)u}{\partial \rho}\right|_{\rho=0} \\ &= -R_b^n \left.\frac{\partial \exp([\rho]_\times^T)u}{\partial \rho}\right|_{\rho=0} \\ &= -R_b^n[u]_\times.\end{aligned} \tag{2.73}$$

Much of the same computation is done for the SO(3) related to the marker rotation acted on $\mathbb{R}^3$. The following equations express a rotation of the marker and an infinitesimally small rotation around the marker coordinate frame in an incremental way:

$$\begin{aligned}\frac{\partial \exp([\rho_n^m]_\times)u}{\partial \rho_m^n} &= -\left.\frac{\partial \exp([\rho]_\times)R_n^m u}{\partial \rho}\right|_{\rho=0} \\ &= [R_n^m u]_\times.\end{aligned} \tag{2.74}$$

$$\begin{aligned}\frac{\partial \exp([\rho_m^n]_\times)u}{\partial \rho_m^n} &= -\left.\frac{\partial R_m^n \exp([\rho]_\times^T)u}{\partial \rho}\right|_{\rho=0} \\ &= -R_m^n[u]_\times.\end{aligned} \tag{2.75}$$

When an SO(3) is not acted on a vector space (R^3), the mapping from input to output perturbation is used [76]:

$$\exp(\rho) \cdot f(g) = f(\exp(\delta) \cdot g), \tag{2.76}$$

$$\frac{\partial f}{\partial g} = \left.\frac{\partial \rho}{\partial \delta}\right|_{\delta=0}, \tag{2.77}$$

where δ is an input perturbation. Here, a function satisfies $f : so(3) \to SO(3)$. Neither the domain nor the range of $f(\rho)$ is a vector space, and the method shown in (2.72) cannot be

used. However, solving (2.76) for ρ yields an explicit formula below:

$$\rho = \log(f(\exp(\delta) \cdot g) \cdot f(g)^{-1}). \tag{2.78}$$

The results of (2.77) and (2.78) yields the formula for the differentiation of the function f as follows:

$$\frac{\partial f(x)}{\partial x} = \left[\frac{\partial}{\partial \rho}\right]_{\rho=0} \left(\log(f(x+\rho) \cdot f(x)^{-1})\right). \tag{2.79}$$

Note that the groups' multiplication ($\cdot$) and inversion (f^{-1}) are identical to matrix multiplication and inversion [77]. For example,

$$\frac{\partial \exp([\rho_n^b]_\times)}{\partial \rho_n^b} = \left[\frac{\partial}{\partial \rho}\right]_{\rho=0} \log(\exp([\rho]_\times) \exp([\rho_n^b]_\times) \exp([\rho_n^b]_\times)^{-1}) \tag{2.80}$$

$$= I_3. \tag{2.81}$$

2.4 Extended Kalman filter

The system equations considered in an extended Kalman filter (EKF) are given as follows:

$$\dot{x} = f(x, w, t), \tag{2.82}$$

$$y = h(x, v, t). \tag{2.83}$$

The EKF is the Kalman filter applied to a linearized system. The linearized system is obtained by linearizing (2.82) and (2.83) at the current estimated states. Let the following equations denote the linearized system:

$$\dot{x}(t) = A(t)x(t) + G(t)w(t), \tag{2.84}$$

$$y(t) = C(t)x(t) + v(t). \tag{2.85}$$

For a discrete-time system, the following notations are used:

$$x_k = F_{k-1}x_{k-1} + G_{k-1}w_{k-1}, \tag{2.86}$$

$$y = H_k x_k + v_k, \tag{2.87}$$

where $w \sim N(0, Q)$, $v \sim N(0, \Sigma)$, and k is a time index.

2.4.1 Propagation

In a propagation phase, also known as a prediction phase, both the estimated states and error covariance matrix are propagated using the known statistics of the system. The derivative of the state vector is computed using a nonlinear system equation as follows:

$$\dot{\hat{x}}(t) = f(\hat{x}(t)), \tag{2.88}$$

and the current estimation is obtained by using numerical integration methods. The way of integration varies for each state, and the details are described in Chapter 3. The estimated error covariance matrix is defined as follows:

$$P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]. \tag{2.89}$$

Equation (2.89) can be expanded to understand how the covariance of the error changes with time.

$$(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T \tag{2.90}$$

$$=[(F_{k-1}x_{k-1} + G_{k-1}w_{k-1}) - (F_{k-1}\hat{x}_{k-1})] \tag{2.91}$$
$$[(F_{k-1}x_{k-1} + G_{k-1}w_{k-1}) - (F_{k-1}\hat{x}_{k-1})]^T$$

$$=[F_{k-1}(x_{k-1} - \hat{x}_{k-1}) + G_{k-1}w_{k-1}] \tag{2.92}$$
$$[F_{k-1}(x_{k-1} - \hat{x}_{k-1}) + G_{k-1}w_{k-1}]^T$$

$$=F_{k-1}(x_{k-1} - \hat{x}_{k-1})(x_{k-1} - \hat{x}_{k-1})^T F_{k-1}^T + F_{k-1}(x_{k-1} - \hat{x}_{k-1})w_{k-1}^T G_{k-1}^T \cdots \tag{2.93}$$
$$+ G_{k-1}w_{k-1}(x_{k-1} - \hat{x}_{k-1})^T F_{k-1}^T + G_{k-1}w_{k-1}w_{k-1}^T G_{k-1}^T.$$

Remember $(x_{k-1} - \hat{x}_{k-1})$ and (w_{k-1}) are not correlated; therefore, the following is true:

$$E[(x_{k-1} - \hat{x}_{k-1})w_{k-1}] \tag{2.94}$$

$$=E[x_{k-1} - \hat{x}_{k-1}]E[w_{k-1}] \tag{2.95}$$

$$=E[x_{k-1} - \hat{x}_{k-1}] \cdot 0 \tag{2.96}$$

$$=0. \tag{2.97}$$

Finally, the error covariance matrix is expressed as follows using (2.89), (2.90), and (2.94):

$$P_k = F_{k-1}P_{k-1}F_{k-1}^T + G_{k-1}Q_{k-1}G_{k-1}^T. \tag{2.98}$$

This final outcome, which propagates the error covariance matrix, is called the discrete-time Lyapunov equation, or the Stein equation [78].

2.4.2 Covariance matrix in continuous-time

If the process noise $w(t)$ is assumed to be continuous-time white noise with a covariance of $Q_c(t)$, this can be expressed as following:

$$E[w(\tau)w^T(\alpha)] = Q_c(\tau)\delta(\tau - \alpha). \tag{2.99}$$

For the time-varying continuous-time system (2.84), the matrices $A(t)$ and $G(t)$ are assumed to be constant for the time $t \in [t_{k-1}, t_k]$. Then, the state at $t = t_k$ is given as

$$\begin{aligned} x(t_k) &= e^{A(t_k - t_{k-1})}x(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}G(\tau)w(\tau)d\tau \\ &= e^{A(\delta t)}x_{k-1} + \int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}G(\tau)w(\tau)d\tau \\ &= F_k x_{k-1} + \int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}G(\tau)w(\tau)d\tau, \end{aligned} \tag{2.100}$$

where $F_k = e^{A\delta t}$ and $\delta t = t_k - t_{k-1}$. The error covariance matrix of this state can be computed in the same way the error covariance matrix of the discrete-time system is obtained,

$$\begin{aligned} P_k &= F_{k-1}P_{k-1}F_{k-1}^T + E\left[\left(\int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}G(\tau)w(\tau)d\tau\right)\left(\int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}G(\tau)w(\tau)d\tau\right)^T\right] \\ &= F_{k-1}P_{k-1}F_{k-1}^T + \int_{t_{k-1}}^{t_k}\int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}G(\tau)E[w(\tau)w(\tau)^T]G(\tau)^T e^{A^T(t_k-\alpha)}d\tau d\alpha. \end{aligned} \tag{2.101}$$

The Dirac delta function satisfies $\delta(\tau - \alpha) = 0$ except at $\delta = \alpha$, and the area drawn by this function is one. Thus,

$$\begin{aligned} &\int f(\tau)\delta(\tau-\alpha)d\tau \\ =&\int f(\alpha)\delta(\tau-\alpha)d\tau \\ =&f(\alpha)\int \delta(\tau-\alpha)d\tau \\ =&f(\alpha)\cdot 1 \\ =&f(\alpha). \end{aligned} \tag{2.102}$$

Using this property, the error covariance matrix (2.101) is expressed as following:

$$P_k = F_{k-1}P_{k-1}F_{k-1}^T + \int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}Q_c(\tau)e^{A^T(t_k-\tau)}d\tau. \tag{2.103}$$

In general, computing the second term of (2.103) is difficult. However, when a time step δt is small, $t_k - \tau$ is also small because $\tau \in [t_{k-1}, t_k]$. Therefore, the matrix exponential is approximated to $e^{A(t_k-\tau)} \approx I$. In that case,

$$\begin{aligned} Q_{k-1} &= \int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}Q_c(\tau)e^{A^T(t_k-\tau)}d\tau \\ &\approx \int_{t_{k-1}}^{t_k} Q_c(\tau)d\tau \\ &\approx Q_c(t_k)\delta t. \end{aligned} \tag{2.104}$$

This relationship between the discrete-time and continuous-time covariance is utilized to propagate the error covariance matrix in the prediction phase, where the dynamics of vehicles and sensors is expressed in continuous-time.

2.4.3 Measurement update

In the measurement update phase of the EKF, estimated states and their error covariance are updated using the following formulas:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H_k\hat{x}_k^-), \tag{2.105}$$

$$P_k^+ = P_k^- - K_k H_k P_k^-, \tag{2.106}$$

where K is the Kalman filter gain calculated as follows:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + \Sigma_k)^{-1}. \tag{2.107}$$

The Kalman filter achieves an optimal estimation in the criterion that the sum of the variances of the estimation errors at time k is minimized. Therefore, the cost function J_k of the optimization problem is defined as follows:

$$\begin{aligned} J_k &= E[(x_{1,k} - \hat{x}_{1,k}^+)^2] + \cdots + E[(x_{n,k} - \hat{x}_{n,k}^+)^2] \\ &= E[e_{x_1,k}^{+2} + \cdots + e_{x_n,k}^{+2}] \\ &= E[e_k^{+T} e_k^+] \\ &= E[\mathrm{Tr}(e_k^+ e_k^{+T})] \\ &= \mathrm{Tr} P_k^+, \end{aligned} \tag{2.108}$$

where

$$e_k^+ = \begin{bmatrix} e_{x_1,k}^+ \\ e_{x_2,k}^+ \\ \vdots \\ e_{x_n,k}^+ \end{bmatrix} = \begin{bmatrix} x_{1,k} - \hat{x}_{1,k}^+ \\ x_{2,k} - \hat{x}_{2,k}^+ \\ \vdots \\ x_{n,k} - \hat{x}_{n,k}^+ \end{bmatrix}. \tag{2.109}$$

The mean of *a posteriori* estimation errors can be expressed using *a priori* errors as follows:

$$\begin{aligned} E[e_k^+] &= E[x_k - \hat{x}_k^+] \\ &= E[x_k - \hat{x}_k^- - K_k(z_k - H_k\hat{x}_k^-)] \\ &= E[e_k^- - K_k(H_k x_k + \sigma v_k - H_k\hat{x}_k^-)] \\ &= E[e_k^- - K_k H_k(x - \hat{x}_k^-) - K_k\sigma v_k] \\ &= E[e_k^- - K_k H_k e_k^- - K_k\sigma v_k] \\ &= (I - K_k H_k)E[e_k^-] - K_k E[\sigma v_k]. \end{aligned} \tag{2.110}$$

Using (2.110), the *a posteriori* error covariance matrix is expressed as following:

$$P_k^+ = E[e_k^+ e_k^{+T}] \tag{2.111}$$

$$= E[\big((I - K_k H_k)e_k^- - K_k\sigma v_k\big)\big((I - K_k H_k)e_k^- - K_k\sigma v_k\big)^T]$$

$$= (I - K_k H_k)E[e_k^- e_k^{-T}](I - K_k H_k)^T - K_k E[\sigma v_k e_k^{-T}](I - K_k H_k)^T \cdots \tag{2.112}$$

$$\cdots - (I - K_k H_k)E[e_k^- v_k\sigma^T]K_k^T + K_k E[\sigma v_k v_k\sigma^T]K_k^T \tag{2.113}$$

$$= (I - K_k H_k)P_k^-(I - K_k H_k)^T + K_k\Sigma_k K_k^T. \tag{2.114}$$

The transformation from (2.112) and (2.113) to (2.114) holds since the *a priori* estimation error e_k^- and measurement error σv_k are not correlated. The K matrix that minimizes the cost function (2.108) makes the following derivative zero:

$$\frac{\partial J_k}{\partial K_k} = 2(I - K_k H_k)P_k^-(-H_k^T) + 2K_k\Sigma_k \ (\because (2.49)). \tag{2.115}$$

From (2.115),

$$
\begin{aligned}
&(I - K_k H_k) P_k^- H_k^T - K_k \Sigma = 0 \qquad (2.116)\\
\Rightarrow &K_k \Sigma_k = (I - K_k H_k) P_k^- H_k^T \\
\Rightarrow &K_k (\Sigma_k + H_k P^- H_k^T) = P_k^- H_k^T \\
\Rightarrow &K_k = P_k^- H_k^T (H_k P_k^- H_k^T + \Sigma_k)^{-1}.
\end{aligned}
$$

When this Kalman filter gain is substituted for (2.110), the simplified form (2.106) is obtained.

2.5 Particle filter

This section introduces useful facts of random variables and derives the particle filter.

2.5.1 Random variables

A random variable is defined as a functional mapping from a set of experimental outcomes to a set of real numbers [79]. The probability density function (PDF) of a random variable X (a capital letter is used to describe the random variable mapped from a real number, which is denoted as a corresponding small letter) is defined as follows:

$$p_X(x) = P(X \leq x), \qquad (2.117)$$

where $P(A)$ denotes the probability of event A. The joint PDF of X_1 and X_2 is expressed as $p_{X_1,X_2}(X_1, X_2)$, and the conditional PDF of X_1 and X_2 is expressed as $p_{X_1|X_2}(X_1|X_2)$. The subscripts of $p(\cdot)$ are dropped when the random variables associated with the PDF are clear from the context. Two important theorems about the joint and conditional PDF are

$$p(x_1|x_2) = \frac{p(x_1, x_2)}{p(x_2)}, \qquad (2.118)$$

and

$$p(x_1|x_2) = \int_{-\infty}^{\infty} p(x_1|x_3)p(x_3|x_2)dx_3. \tag{2.119}$$

The latter equation is often called the Chapman-Kolmogorov equation [80].

Bayes rule

Another important theory to connect prior knowledge to observation is the Bayes rule [81]. Bayes rule of PDFs is described as follows:

$$p(x_1|x_2) = \frac{p(x_2|x_1)p(x_1)}{p(x_2)} = \frac{p(x_2|x_1)p(x_1)}{\int p(x_2|x_1)p(x_1)dx_1}. \tag{2.120}$$

2.5.2 Derivation of particle filter

The goal of the particle filter is to estimate $p(x_k|z_{1:k})$, which is the density of the states conditioned on the sequence of the measurements. The system is the nonlinear system described in the previous section. Once $p(x_k|z_{1:k})$ is computed, the final output of the filter is determined in an appropriate way, depending on the problem. From the Bayes' rule:

$$p(x_k|z_{1:k}) = \frac{p(z_{1:k}|x_k)}{p(z_{1:k})}p(x_k). \tag{2.121}$$

First of all, $z_{1:k}$ is the expression for sequence of measurements $z_1, z_2, z_3, \ldots, z_k$. Separate these measurements to $z_{1:k-1}$ and z_k. By doing so, (2.118) is expressed as follows:

$$\begin{aligned} \frac{p(z_{1:k}|x_k)}{p(z_{1:k})} &= \frac{p(z_k, z_{1:k-1}|x_k)}{p(z_k, z_{1:k-1})} \\ &= \frac{p(x_k, z_k, z_{1:k-1})}{p(x_k)p(z_k, z_{1:k-1})}. \end{aligned} \tag{2.122}$$

Use (2.118) again to transform the marginal PDF $p(x_k)$ as follows:

$$\begin{aligned} p(x_k) &= \frac{p(z_{1:k-1}, x_k)}{p(z_{1:k-1}|x_k)} \\ &= \frac{p(x_k, z_{1:k-1})}{p(z_{1:k-1}|x_k)} \\ &= \frac{p(x_k|z_{1:k-1})p(z_{1:k-1})}{p(z_{1:k-1}|x_k)}. \end{aligned} \tag{2.123}$$

Substitute (2.122) and (2.123) for (2.121), and the following equation is obtained:

$$p(x_k|z_{1:k}) = \frac{p(x_k, z_k, z_{1:k-1})}{p(x_k)p(z_k, z_{1:k-1})} \frac{p(x_k, z_{1:k-1})p(z_{1:k-1})}{p(z_{1:k-1})p(z_{1:k-1}|x_k)}. \tag{2.124}$$

Multiply both the numerator and denominator of the obtained equation by the joint PDF of states and measurements, and the following is obtained:

$$\begin{aligned} p(x_k|z_{1:k}) &= \frac{p(x_k, z_k, z_{1:k-1})}{p(x_k)p(z_k, z_{1:k-1})} \frac{p(x_k, z_{1:k-1})p(z_{1:k-1})}{p(z_{1:k-1})p(z_{1:k-1}|x_k)} \frac{p(x_k, z_k)}{p(x_k, z_k)} \\ &= \frac{p(z_{1:k-1}, x_k, z_k)}{p(x_k, z_k)} \frac{p(z_k, x_k)}{p(x_k)} \frac{p(x_k, z_{1:k-1})}{p(z_{1:k-1})} \frac{p(z_{1:k-1})}{p(z_k, z_{1:k-1})} \frac{1}{p(z_{1:k-1}|x_k)} \\ &= \frac{p(z_{1:k-1}|x_k, z_k)p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})p(z_{1:k-1}|x_k)}. \end{aligned} \tag{2.125}$$

Note that the measurement z_k is a function of x_k, and therefore, $p(z_{1:k-1}|x_k, z_k) = p(z_{1:k-1}|x_k)$. This fact simplifies (2.125) as follows:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})}, \tag{2.126}$$

where

$$p(z_k|z_{1:k-1}) = \int p(z_k, x_k|z_{1:k-1})dx_k \tag{2.127}$$

$$= \int p(z_k|x_k, z_{1:k-1})p(x_k|z_{1:k-1})dx_k \tag{2.128}$$

$$= \int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k. \tag{2.129}$$

Note that the transformation from (2.128) to (2.129) uses the fact that the following conditional PDF can eliminate the preceding measurements as follows:

$$p(z_k|x_k, z_{1:k-1}) = p(z_k|x_k), \tag{2.130}$$

since measurements z are independent. By substituting (2.129) for (2.126), the final version of the *a posteriori* density is obtained as follows:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{\int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k}. \tag{2.131}$$

The *a priori* PDF is obtained in much the same way:

$$p(x_k|z_{1:k-1}) = \int p(x_k, x_{k-1}|z_{1:k-1})dx_{k-1} \tag{2.132}$$

$$= \int p(x_k|x_{k-1}, z_{1:k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{2.133}$$

$$= \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1}. \tag{2.134}$$

This is the way to obtain *a priori* density. $p(x_k|x_{k-1})$ is obtained through the known system dynamics, and $p(x_{k-1}|z_{1:k-1})$ is the *a posteriori* density in the previous step. Finally, (2.131) and (2.134) achieves the recursive Bayesian state estimation, and the filter that numerically implements these two steps is called the particle filter.

CHAPTER 3

VISION-AIDED NAVIGATION DESIGN

This chapter presents algorithms to estimate the states of vehicles and targets. The algorithms used in this thesis are a standard extended Kalman filter (EKF), extended Kalman particle filter (PF-EKF), and Rao-Blackwellized particle filter (RBPF). All these algorithms are used in the paradigm of simultaneous localization and mapping (SLAM) using visual measurements, which is also known as visual SLAM. The relative position and attitude measurements produced by vision are mapped in navigation coordinate frame. Other onboard sensors are also integrated in the algorithms. This chapter first introduces an EKF to estimate the states of vehicle and sensor biases. This part of the EKF is used across three different filter designs. The EKF, or a particle when multiple-hypothesis approaches are used, is propagated using an inertial measurement unit (IMU) and the estimated states of self. Other sensors except vision are utilized to update the estimated states in the Kalman equations. In the following section, marker-aided EKF is introduced. By estimating the states of the vehicle and the marker simultaneously, the estimation errors are reduced and the system becomes robust to the GPS failures. Next Section introduces PF-EKF. This Section describes how to combine the particle filter techniques in the framework of EKF. Also, PF-EKF does not produce a single navigation solution; instead, what it produces is the distribution of the estimated states. The way to compute the navigation solution from the distribution is also explained in this section. The final section of this chapter presents the RBPF algorithm. The marker states are estimated in a complete particle filter (PF) framework. The marginalization of states estimated in EKF and PF frameworks are described. In all filter implementations, images are captured in a monocular camera, and the positions (and optionally, the orientations) of markers are extracted from images. The measurements of markers are used to take the possibility of the false positives and negatives

into consideration. A normalized square norm of the residual (Mahalanobis distance) sets an adaptive threshold to judge whether or not a new observation comes from the target the particle is tracking.

3.1 Extended Kalman filter to estimate vehicle states

This section presents an extended Kalman filter (EKF) to estimate the state of a vehicle. This algorithm is used even before observing a visual target. Also, this part of the EKF is utilized for PF-EKF and RBPF. The attitude, position, and velocity of the vehicle, and the biases of the gyroscope and accelerometer are estimated in this algorithm. Let x denote the state vector of the estimator. Then, it is expressed as follows:

$$\hat{x} = \begin{bmatrix} \hat{\rho}_b^n & \hat{p}_{nb}^n & \hat{v}_{nb}^n & \hat{b}_{\text{gyro}} & \hat{b}_{acc} \end{bmatrix} \in \mathbb{R}^{15}, \tag{3.1}$$

where ρ_b^n is the axis-angle representing a rotation from body to navigation frames. That means the vector ρ can be interpreted as a rotation around the axis $\rho/||\rho||$ by $||\rho||$ radian. p_{nb}^n stands for the position of the vehicle with respect to the origin of the navigation frame resolved in the navigation frame. The vector is measured with respect the coordinate frames represented by the subscripts. Figure 3.1 shows examples of the notations. The superscript indicates the coordinate frame, in which the vector is resolved. For example, p_{bm}^b is the vector from O_b to O_m expressed in the body frame. This vector can be expressed in the camera coordinate frame using the rotation matrix from the body to camera frames as follows: $p_{bm}^c = R_b^c p_{bm}^b$. Much the same expression is used for v_{nb}^n, which is the velocity of the vehicle with respect to the navigation frame expressed in the navigation frame. Here, the navigation frame is a local north-east-down (NED) coordinate frame. Filter solutions are solved with respect to the NED frame, and without loss of generality, the NED frame is regarded as an inertial frame when expressing dynamics. The biases of the on-board sensors are always expressed in the sensor coordinates. In this section, an IMU is assumed

to be located on the origin of the body coordinate, which is identical to the center of gravity (C.G.) of body. The case when an IMU is not located on C.G. is covered by Appendix B. b_{gyro} is the bias of the gyroscope measurements; thus, it is the bias of the angular velocity measurements of the body. b_{acc} is the bias of the accelerometer measurements. Therefore, it is the bias of the specific acceleration measurements of the body.

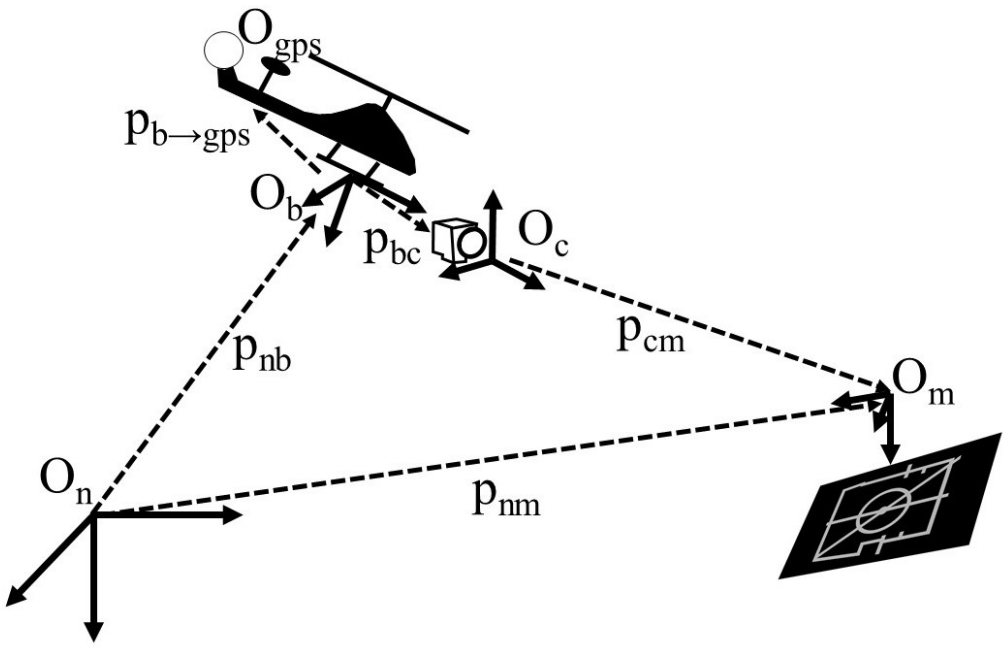


Figure 3.1: Figure explains the coordinate frames and the notations used in this Chapter.

3.1.1 Sensor model

In this EKF model, five sensors are primarily used: gyroscope, accelerometer, global positioning system (GPS), magnetometer, and barometric pressure sensor. An inertial measurement unit (IMU) is composed of a gyroscope and an accelerometer. Let z_{gyro}, z_{acc}, z_{gps}, z_{mag}, and z_{baro} be the measurements of the gyroscope, accelerometer, GPS, magnetometer, and barometric pressure sensor, respectively. Firstly, the measurement of a gyroscope is modeled as follows:

$$z_{\text{gyro}} = \omega_{nb}^{b} + b_{\text{gyro}} + \sigma_{\text{gyro}}\nu, \tag{3.2}$$

where ω_{nb}^b is the angular velocity of the body, $\sigma_{\text{gyro}}^2 \in \mathbb{R}^3$ is the variance of the measurement noise of the gyroscope, and ν is the random variable that satisfies $\nu \sim N(0,1)$, which denotes a normal-distributed zero-mean random variable with the variance of one. The same IMU model is used in many studies, and examples include [82] and [83]. Note that all the biases are modeled with the first-order autoregressive random variable, AR(1), and the model described as follows:

$$b(t+dt) = e^{-\lambda dt} b(t) + \sqrt{1 - e^{-2\lambda dt}} \sigma \nu. \tag{3.3}$$

Note that the noise variance for the biases, modeled as an exponentially correlated fixed-variance first-order Markov process is:

$$Q = \left(1 - e^{-2dt/\tau}\right) \sigma^2, \tag{3.4}$$

where τ is the correlation time (14.2.6 of [61]). The continuous model of this can be expressed as follows:

$$\dot{b} = -\lambda b + \nu_c, \tag{3.5}$$

where ν_c is the corresponding noise in continuous time. This way, biases are modeled as exponentially correlated fixed-variance first-order Markov processes (see Appendix B.4.3 of [61]). When higher-order IMU errors need to be modeled, [71] provides a rigorous model. In some literature, biases are modeled as constant [84], [65]. In [69], when there is no good information to determine the prediction model, the constant bias model is used. These approaches are reasonable during a short time interval, but whenever possible, the biases should be modeled as time-dependent variables. Next, the measurement of an accelerometer is

$$z_{acc} = R_n^b (a_{nb}^n - g^n) + b_{acc} + \sigma_{acc} \nu, \tag{3.6}$$

where a^n_{nb} is the acceleration of the body with respect to the navigation frame expressed in the navigation frame. g^n is the gravitational acceleration expressed in navigation frame. σ_{acc} is the standard deviation of the measurement noise of the accelerometer. The IMU error characteristics for different grades are summarized in [85]. An IMU is typically located at the center of gravity (C.G.), but the GPS antenna may not or cannot be placed at C.G. Let $p^b_{b\to\text{gps}}$ denote the position of the GPS antenna with respect to the body center resolved in the body coordinate. Then, the GPS measurement can be expressed as follows:

$$z_{\text{gps}} = \begin{bmatrix} p^n_{nb} \\ v^n_{nb} \end{bmatrix} + \begin{bmatrix} R^n_b p^b_{b\to\text{gps}} \\ R^n_b(\omega^b_{nb} \times p^b_{b\to\text{gps}}) \end{bmatrix} + \sigma_{\text{gps}}\nu. \tag{3.7}$$

Magnetometer measurements can be expressed as follows:

$$z_{mag} = R^b_n B^n + B^b_{\text{dist}} + \sigma_{mag}\nu, \tag{3.8}$$

where B^n is the magnetic field expressed in the navigation frame, and B^b_{dist} is a body-fixed expression for the local magnetic disturbance. In this model, the magnetic field is fixed based on the coordinates of the navigation frame. For instance, the national centers for environmental information (NOAA) provide a tool[1] to compute the magnetic field with the given latitude and longitude. In this tool, the magnetic field expressed in the NED frame is $B^n = \begin{bmatrix} 22,678.0 & -2,029.1 & 43,600.5 \end{bmatrix} (nT)$ in Atlanta, where latitude and longitude are 33.749N (deg) and 84.388W (deg). The standard deviation of the measurement noise σ_{mag} is usually low for magnetometer readings [69], but the B^b_{dist} can be significant, especially if the magnetometer is placed near the power wires to the motors. In the work here, B^b_{dist} is known, and measurements are compensated for it before computing the measurement residuals. The effects of uncertainty of the local disturbance is discussed in Appendix D.

[1] `https://www.ngdc.noaa.gov/geomag-web/`

Finally, the barometric pressure sensor measures the altitude of the body as follows:

$$z_{baro} = p^n_{nb,z} + \sigma_{baro}\nu, \tag{3.9}$$

where $p^n_{nb,z}$ represents the z component of the p^n_{nb} vector, which indicates the altitude of the body in NED frame.

3.1.2 Propagation

Propagation, also known as prediction, of the estimated states is done by using the IMU measurements. The states are propagated using nonlinear equations, and the estimated error covariance matrix is propagated with the discrete Lyapunov equation. The estimated angular velocity of the body is computed as follows

$$\hat{\omega}^b_{nb} = z_{\text{gyro}} - \hat{b}_{\text{gyro}}, \tag{3.10}$$

and estimated specific acceleration applied to a body is:

$$\hat{s}^b_{nb} = z_{acc} - \hat{b}_{acc}. \tag{3.11}$$

The known local gravity models provides the estimated acceleration of the body as follows:

$$\hat{a}^n_{nb} = \hat{R}^n_b \hat{s}^b_{nb} + g^n. \tag{3.12}$$

The velocity is propagated with the first-order-forward-Euler, and the position is propagated with the second-order-forward-Euler method:

$$\hat{p}^n_{nb}(t + dt) = \hat{p}^n_{nb}(t) + \hat{v}^n_{nb}(t)dt + \frac{1}{2}\hat{a}^n_{nb}(t)dt^2, \tag{3.13}$$

$$\hat{v}^n_{nb}(t+dt) = \hat{v}^n_{nb}(t) + \hat{a}^n_{nb}(t)dt. \tag{3.14}$$

The attitude of the vehicle is stored as a rotation matrix and propagated as follows using matrix exponential:

$$\hat{R}^n_b(t+dt) = \hat{R}^n_b(t)e^{[\hat{w}^b_{nb}]_\times dt}. \tag{3.15}$$

Although this propagation uses a full exponential mapping, this integration can be approximated to the first order forward Euler. The time derivative of the rotation matrix is expressed as $\dot{R}^n_b = R^n_b[w^b_{nb}]_\times$ (2.37); therefore, $R^n_b(t+dt) = R^n_b(t) + \dot{R}^n_b dt = R^n_b(t) + R^n_b(t)[w^b_{nb}]_\times dt = R^n_b(t)(I + [w^b_{nb}]_\times dt)$. This is equivalent to the first-order approximation of the exponential matrix $e^{[w]_\times dt} \approx I + [w]_\times dt$. Finally, biases are the first-order autoregressive, AR(1), and they are propagated as follows:

$$\hat{b}_{\text{gyro}}(t+dt) = e^{-\lambda_{\text{gyro}}dt}\hat{b}_{\text{gyro}}(t), \tag{3.16}$$

$$\hat{b}_{acc}(t+dt) = e^{-\lambda_{acc}dt}\hat{b}_{acc}(t). \tag{3.17}$$

The propagation of the estimated error covariance matrix requires the Jacobian matrix:

$$A = \frac{\partial f(x)}{\partial x}\|_{x=\hat{x}}. \tag{3.18}$$

By using the estimated parameters, this Jacobian matrix can be expressed as follows:

$$A = \left[\begin{array}{c|c|c|c} -[\hat{\omega}^b_{nb}]_\times & & -I_3 & \\ \hline & I_3 & & \\ \hline -\hat{R}^n_b[\hat{s}^b_{nb}]_\times & & & -\hat{R}^n_b \\ \hline & & -\lambda_{\text{gyro}}I_3 & \\ \hline & & & -\lambda_{acc}I_3 \end{array}\right], \tag{3.19}$$

Non-trivial entries of the Jacobian matrix are derived as follows: Firstly, from (2.36),

$$\dot{\rho}_b^n \approx R_b^n w_{nb}^b \tag{3.20}$$

$$= \exp([\rho_b^n]_\times) w_{nb}^b \tag{3.21}$$

$$\approx (I + [\rho_b^n]_\times) w_{nb}^b \tag{3.22}$$

Note that the Jacobian matrix needs to be evaluated locally at the body coordinate frame as explained in Subsection 2.3.2. Using the results of Subsection 2.3.2, the following entries are derived:

$$\frac{\partial \dot{\hat{\rho}}_b^n}{\partial \hat{\rho}_b^n} = \left. \frac{\partial((I + [\rho]_\times)\hat{w}_{nb}^b}{\partial \rho} \right|_{\rho=0} (\because (3.22)) \tag{3.23}$$

$$= \left. \frac{\partial(-[\hat{w}_{nb}^b]_\times \rho)}{\partial \rho} \right|_{\rho=0} (\because (2.11)) \tag{3.24}$$

$$= -[\hat{w}_{nb}^b]_\times, \tag{3.25}$$

$$\frac{\partial \dot{\hat{\rho}}_b^n}{\partial \hat{b}_{\text{gyro}}} = \left. \frac{\partial((I + [\rho]_\times)\hat{w}_{nb}^b}{\partial \hat{b}_{\text{gyro}}} \right|_{\rho=0} (\because (3.22)) \tag{3.26}$$

$$= \left. \frac{\partial((I + [\rho]_\times)(z_{\text{gyro}} - \hat{b}_{\text{gyro}})}{\partial \hat{b}_{\text{gyro}}} \right|_{\rho=0} (\because (3.10)) \tag{3.27}$$

$$= -\left. (I + [\rho]_\times) \right|_{\rho=0} \tag{3.28}$$

$$= -I, \tag{3.29}$$

$$
\begin{aligned}
\frac{\partial \dot{\hat{v}}_{nb}^{b}}{\partial \hat{\rho}_{b}^{n}} &= \frac{\partial \hat{a}_{nb}^{b}}{\partial \hat{\rho}_{b}^{n}} \\
&= \frac{\partial (\hat{R}_{b}^{n}(z_{\mathrm{acc}} - \hat{b}_{\mathrm{acc}}) + g^{n})}{\partial \hat{\rho}_{b}^{n}} \ (\because (3.12)) \\
&= \frac{\partial (\hat{R}_{b}^{n} \hat{s}_{nb}^{b} + g^{n})}{\partial \hat{\rho}_{b}^{n}} \\
&= \frac{\partial (\hat{R}_{b}^{n} \hat{s}_{nb}^{b})}{\partial \hat{\rho}_{b}^{n}} \\
&= -\hat{R}_{b}^{n} [\hat{s}_{nb}^{b}]_{\times} \ (\because \ (2.73)),
\end{aligned}
\tag{3.30}
$$

and finally,

$$
\begin{aligned}
\frac{\partial \dot{\hat{v}}_{nb}^{b}}{\partial \hat{b}_{\mathrm{acc}}} &= \frac{\partial \hat{a}_{nb}^{b}}{\partial \hat{b}_{\mathrm{acc}}} \ (\because (3.12)) \\
&= \frac{\partial (\hat{R}_{b}^{n}(z_{\mathrm{acc}} - \hat{b}_{\mathrm{acc}}) + g^{n})}{\partial \hat{b}_{\mathrm{acc}}} \\
&= \frac{\partial (\hat{R}_{b}^{n}(-\hat{b}_{\mathrm{acc}}))}{\partial \hat{b}_{\mathrm{acc}}} \\
&= -\hat{R}_{b}^{n}.
\end{aligned}
\tag{3.31}
$$

The Jacobian matrix in discrete-time is approximated as follows:

(This approximation is valid only when a time step dt is sufficiently small.)

$$
F = I + Adt. \tag{3.32}
$$

The *a priori* estimated error covariance matrix is propagated with the discrete-time Lyapunov equation (2.98) as follows:

$$
P_{k}^{-} = F_{k} P_{k}^{+} F_{k}^{T} + G_{k} Q G_{k}^{T}, \tag{3.33}
$$

where the process noise covariance matrix Q is computed as follows:

$$Q = \left[\begin{array}{c|c|c|c} \Sigma_{\text{gyro}}dt & & & \\ \hline & & & \\ \hline & \Sigma_{acc}dt & & \\ \hline & & 2\lambda_{\text{gyro}}\Sigma_{\text{gyro}}dt & \\ \hline & & & 2\lambda_{acc}\Sigma_{acc}dt \end{array}\right], \ (\because (2.104)) \tag{3.34}$$

and

$$G_k = \left[\begin{array}{c|c|c|c|c} I & & & & \\ \hline & I & & & \\ \hline & & \hat{R}^n_{b,k} & & \\ \hline & & & I & \\ \hline & & & & I \end{array}\right]. \tag{3.35}$$

Note that all the sensor measurement covariances are expressed in the sensor frames. Since the estimated velocity is resolved in the NED frame and not the sensor frame, the above conversion (3.35) is necessary.

3.1.3 Update estimated states with sensors

All sensors except the accelerometer and gyroscope are used to update estimated states in a discrete sense. Each sensor has a different output matrix H:

$$H = \frac{\partial y}{\partial x}\|_{x=\hat{x}}. \tag{3.36}$$

For a GPS (provided that a GPS measures the position and velocity),

$$H_{\text{gps}} = \left[\begin{array}{c|c|c|c|c} -\hat{R}^n_b[p^b_{b\to\text{gps}}]_\times & I_3 & & & \\ \hline -\hat{R}^n_b[\hat{\omega}^b_{nb} \times p^b_{b\to\text{gps}}]_\times & & I_3 & \hat{R}^n_b[p^b_{b\to\text{gps}}]_\times & \end{array}\right], \tag{3.37}$$

where the outputs and the residuals of GPS are expressed as follows:

$$y_{\text{gps}} = \begin{bmatrix} y_{\text{gps},p} \\ y_{\text{gps},v} \end{bmatrix} = \begin{bmatrix} \hat{p}^n_{nb} \\ \hat{v}^n_{nb} \end{bmatrix} + \begin{bmatrix} \hat{R}^n_b p^b_{b\to\text{gps}} \\ \hat{R}^n_b(\hat{\omega}^b_{nb} \times p^b_{b\to\text{gps}}) \end{bmatrix}, \tag{3.38}$$

$$\epsilon_{\text{gps}} = z_{\text{gps}} - y_{\text{gps}}. \tag{3.39}$$

The non-trivial entries of the Jacobian matrix are computed as follows:

$$\begin{aligned} \frac{\partial y_{\text{gps,v}}}{\partial \hat{b}_{\text{gyro}}} &= \frac{\partial(\hat{v}^n_{nb} + \hat{R}^n_b(\hat{w}^b_{nb} \times p^b_{b\to\text{gps}}))}{\partial \hat{b}_{\text{gyro}}} \\ &= \frac{\partial(\hat{v}^n_{nb} - \hat{R}^n_b(p^b_{b\to\text{gps}} \times \hat{w}^b_{nb}))}{\partial \hat{b}_{\text{gyro}}} \ (\because (2.11)) \\ &= \frac{\partial(\hat{v}^n_{nb} - \hat{R}^n_b(p^b_{b\to\text{gps}} \times (z_{\text{gyro}} - \hat{b}_{\text{gyro}})))}{\partial \hat{b}_{\text{gyro}}} \ (\because (3.10)) \\ &= \frac{\partial \hat{R}^n_b [p^b_{b\to\text{gps}}]_\times \hat{b}_{\text{gyro}}}{\partial \hat{b}_{\text{gyro}}} \\ &= \hat{R}^n_b [p^b_{b\to\text{gps}}]_\times. \end{aligned} \tag{3.40}$$

The $\frac{\partial y_{\text{gps},p}}{\partial \hat{\rho}^n_b}$ and $\frac{\partial y_{\text{gps},v}}{\partial \hat{\rho}^n_b}$ are derived using (2.73). For a magnetometer,

$$H_{mag} = \left[\begin{array}{c|c|c|c|c} [\hat{R}^{nT}_b B^n]_\times & & & & \end{array} \right], \tag{3.41}$$

where

$$y_{mag} = \hat{R}^{nT}_b B^n + B^b_{\text{dist}}, \tag{3.42}$$

$$\epsilon_{mag} = z_{mag} - y_{mag}. \tag{3.43}$$

The non-trivial entry is derived as follows:

$$\frac{\partial y_{\text{mag}}}{\partial \hat{\rho}_b^n} = \frac{\partial(\hat{R}_b^{nT} B^n + B_{\text{dist}}^b)}{\partial \hat{\rho}_b^n} \tag{3.44}$$

$$= \frac{\partial(\hat{R}_b^{nT} B^n)}{\partial \hat{\rho}_b^n} \tag{3.45}$$

$$= [\hat{R}_b^{nT} B^n]_\times \ (\because (2.72)). \tag{3.46}$$

For a barometric pressure sensor,

$$H_{baro} = \left[\ \middle| \ \middle| \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \middle| \ \middle| \ \middle| \ \right]. \tag{3.47}$$

Note that the second column of the H matrix above corresponds to the position of the vehicle, and the non-zero entry of the matrix corresponds to the third element of p_{nb}^n, which is the altitude of the vehicle. The outputs of a barometric pressure sensor are simply expressed as follows:

$$y_{baro} = \hat{p}_{nb,z}^n. \tag{3.48}$$

Therefore, the residual is computed as follows:

$$\epsilon_{baro} = z_{baro} - y_{baro}. \tag{3.49}$$

Using these H matrices and residuals, the estimated states and the error covariance matrix are updated.

$$K = P(k)^- H^T (H P(k)^- H^T + \Sigma)^{-1} \tag{3.50}$$

$$P(k)^+ = (I - KH) P(k)^- \tag{3.51}$$

$$x(k)^+ = x^-(k) \oplus dx, \tag{3.52}$$

where $dx = K\epsilon$. Note that the $\oplus$ operator denotes the mean update operation of Kalman filters, which need not satisfy a simple addition of vectors [86]. For example, the attitude of the vehicle is stored as a rotation matrix. The update is not a simple addition but it is expressed as

$$\hat{R}_b^n(k)^+ = \hat{R}_b^n(k)^- e^{[dx_\rho]_\times}, \tag{3.53}$$

where

$$dx = \begin{bmatrix} dx_\rho & dx_p & dx_v & dx_{b_{\text{gyro}}} & dx_{b_{acc}} \end{bmatrix}. \tag{3.54}$$

In the update equations above, H is replaced by either H_{gps}, H_{mag}, or H_{baro} and Σ is replaced by Σ_{gps}, Σ_{mag}, or Σ_{baro}.

3.1.4 Initialization

The initialization of the attitude (direction cosine matrix) is conducted by using the combination of the accelerometer and the magnetometer. Equations (3.6) and (3.8), models of these sensors, are important here. By ignoring the noises of the measurements, the equations below are established:

$$z_{acc} \approx \hat{R}_{n,\text{init}}^b (a_{nb}^n - g^n) + b_{acc}, \tag{3.55}$$

$$z_{mag} \approx \hat{R}_{n,\text{init}}^b B^n. \tag{3.56}$$

Two assumptions are made here: there are no bias of the measurement, and the vehicle has no acceleration at initialization. Then, the equations are used at the initialization:

$$z_{acc} = \hat{R}_{n,\text{init}}^b (-g^n), \tag{3.57}$$

and

$$z_{mag} = \hat{R}^{b}_{n,\text{init}} B^{n}. \tag{3.58}$$

A rotation matrix is a 3×3 matrix, consisting of a total of nine members; however, it only takes three numbers to determine the matrix completely. Each of the magnetometer and accelerometer measurements provides two of three degrees of freedom, and determining the initial rotation matrix becomes an overdetermined problem. One of the simplest algorithms to solve this problem is called *triad method*, which constructs two *triads* of orthonormal unit vectors from two sets of vectors. Note this method only works when the two vectors are not colinear; thus, magnetic vector and gravitational vector can not be parallel. This happens when the observations occur at the magnetic poles (i.e. the North and the South poles.), and since in most of UAV operations are made outside of this condition, the exception is ignored. Here, orthonormal vectors are constructed by accelerometer measurements ($b1$), the vector perpendicular to both accelerometer and magnetometer measurements ($b2$), and the vector perpendicular to b1 and b2 ($b3$). The counterpart of the orthonormal vectors ($n1, n2, n3$) are constructed using the known navigation-frame components. All column vectors of the *triad* matrices are normalized, and they are computed as follows:

$$\begin{bmatrix} b1 & b2 & b3 \end{bmatrix} = \begin{bmatrix} z_{acc} & [z_{acc}]_{\times} z_{mag} & [z_{acc}]_{\times}([z_{acc}]_{\times} z_{mag}) \end{bmatrix}, \tag{3.59}$$

and

$$\begin{bmatrix} n1 & n2 & n3 \end{bmatrix} = \begin{bmatrix} -g^{n} & -[g^{n}]_{\times} B^{n} & -[g^{n}]_{\times}(-[g^{n}]_{\times} B^{n}) \end{bmatrix}. \tag{3.60}$$

Using these matrices, the initial attitude is determined as follows:

$$\begin{bmatrix} z_{acc} & [z_{acc}]_{\times} z_{mag} & [z_{acc}]_{\times}([z_{acc}]_{\times} z_{mag}) \end{bmatrix} = \hat{R}^{b}_{n,\text{init,raw}} \begin{bmatrix} -g^{n} & -[g^{n}]_{\times} B^{n} & -[g^{n}]_{\times}(-[g^{n}]_{\times} B^{n}) \end{bmatrix}. \tag{3.61}$$

Remember that the column vectors in the *triad* matrix of the right-hand-side of (3.61) are orthonormal to each other; therefore, the inverse is obtained through transposing the

matrix:

$$\hat{R}^b_{n,\text{init,raw}} = \begin{bmatrix} z_{acc} & [z_{acc}]_\times z_{mag} & [z_{acc}]_\times([z_{acc}]_\times z_{mag}) \end{bmatrix} \begin{bmatrix} -g^n & -[g^n]_\times B^n & -[g^n]_\times(-[g^n]_\times B^n) \end{bmatrix}^{-1} \tag{3.62}$$

$$= \begin{bmatrix} z_{acc} & [z_{acc}]_\times z_{mag} & [z_{acc}]_\times([z_{acc}]_\times z_{mag}) \end{bmatrix} \begin{bmatrix} -g^n & -[g^n]_\times B^n & -[g^n]_\times(-[g^n]_\times B^n) \end{bmatrix}^{T}. \tag{3.63}$$

This algorithm is introduced in [87], [88], and a numerical example is provided by Chapter 4 of [89], although the example uses a sun sensor and magnetometer. This rotation matrix is not necessarily a proper orthogonal matrix, which is an orthogonal matrix with the determinant of plus one. This matrix is projected to $SO(3)$ using the singular value decomposition as follows:

$$\hat{R}^b_{n,\text{init,raw}} = USV^T, \tag{3.64}$$

$$\hat{R}^b_{n,\text{init}} \leftarrow UMV^T, \tag{3.65}$$

where

$$M = \text{diag}\begin{bmatrix} 1 & 1 & \det(U)\det(V) \end{bmatrix}. \tag{3.66}$$

Since U and V are unitary matrices, the determinant of them are either $+1$ or -1. By multiplying their determinants in (3.65), the final result of the rotation matrix is guaranteed to have a determinant of plus one [90]. The use of the singular value decomposition as shown in [90] is also introduced in [91] with magnetometer and accelerometer measurements. Initial position and velocity are determined by the GPS measurement. Note that the angular velocity is assumed to be zero at initialization:

$$\hat{p}^n_{nb,\text{init}} = z_{\text{gps},p} - R^n_{b,\text{init}} p^b_{b\to\text{gps}}. \tag{3.67}$$

$$\hat{v}^n_{nb,\text{init}} = z_{\text{gps},v} - R^n_{b,\text{init}}(\hat{\omega}^b_{nb,\text{init}} \times p^b_{b\to\text{gps}}) \tag{3.68}$$
$$\approx z_{\text{gps},v} \ (\because \hat{\omega}^b_{nb,\text{init}} = 0).$$

3.2 Visual SLAM with moving visual marker

This section describes the visual SLAM architecture used in this work. Let m denote the marker coordinate. This is used as a superscript and subscript to describe marker-related parameters. In addition to 15-states shown in (3.1), nine more parameters must also be estimated, namely the attitude, position, and velocity of a marker.

$$\hat{x} = \begin{bmatrix} \hat{\rho}^n_b & \hat{p}^n_{nb} & \hat{v}^n_{nb} & \hat{b}_{gyro} & \hat{b}_{acc} & \hat{\rho}^n_m & \hat{p}^n_{nm} & \hat{v}^n_{nm} \end{bmatrix} \in \mathbb{R}^{24}. \tag{3.69}$$

The same notations are used from the previous section. $\hat{\rho}^n_m$ is the estimated attitude of marker expressed in axis-angle with respect to the navigation frame. $\hat{p}^n_{nm}$ is the estimated position of a marker with respect to the navigation frame expressed in the navigation frame. Finally, $\hat{v}^n_{nm}$ is the estimated velocity of a marker with respect to the navigation frame. In addition to the sensors used for the conventional model (gyroscope, accelerometer, GPS, magnetometer, and barometric pressure sensor), vision is added. Also, another coordinate frame is defined, which is the camera frame. Let c denote the camera. The intrinsic parameters (focal length and optical center) and translation and rotation of a camera with respect to a body (R^c_b and p^b_{bc}) are assumed to be known. The use of intrinsic parameters as well as other calibration parameters (e.g. distortion parameters) is explained in Chapter 4.

3.2.1 Sensor model

A detection framework that measures the attitude and the position of a visual marker is used. The visual detection framework outputs the attitude of the marker as a rotation matrix converting the marker to camera frames. Let $\tilde{R}^c_m$ denote measurements of marker attitude;

then, it is modeled:

$$\tilde{R}_m^c = R_m^c e^{[\sigma_{m\rho}\nu]_\times}, \tag{3.70}$$

where σ_m is the standard deviation of the measurement errors written as follows:

$$\sigma_m = \begin{bmatrix} \sigma_{m\rho} \\ \sigma_{mp} \end{bmatrix} \in \mathbb{R}^6. \tag{3.71}$$

This means that a measurement of marker detection is expressed as a six-dimensional vector, $\sigma_{m\rho}$ represents the deviation of the measurement noise regarding the attitude, and σ_{mp} represents the standard deviation of the measurement noise regarding the position. Not to mention, but the measurement error covariance matrix of the marker detection is

$$\Sigma_m^c = E[\sigma_m \sigma_m^T] \in \mathbb{R}^{6\times 6}. \tag{3.72}$$

A measurement of the position of a marker is expressed as:

$$\tilde{p}_{cm}^c = p_{cm}^c + \sigma_{mp}\nu. \tag{3.73}$$

Equations (3.70) and (3.73) are the direct measurement from the vision system. Since the assumption is that the translation and rotation from the body to camera frames are known, the marker measurements may be treated as the rotation and translation from the marker to the body, thus:

$$z_m = \begin{bmatrix} \log(\tilde{R}_m^b) \\ \tilde{p}_{bm}^b \end{bmatrix} = \begin{bmatrix} \log(R_c^b \tilde{R}_m^c) \\ p_{bc}^b + R_c^b \tilde{p}_{cm}^c \end{bmatrix} \in \mathbb{R}^6. \tag{3.74}$$

When it comes to validating the simulation with synthetic data, the synthetic measurements need to be produced. In that case, the following equations are used:

$$R_m^c = R_b^c (R_b^n)^T R_m^n, \tag{3.75}$$

and

$$p^c_{cm} = R^c_b\{R^{nT}_b(p^n_{nm} - p^n_{nb}) - p^b_{bc}\}. \tag{3.76}$$

3.2.2 Propagation

The position of a marker is propagated using the estimated velocity of the marker:

$$\hat{p}^n_{nm}(t + dt) = \hat{p}^n_{nm}(t) + \hat{v}^n_{nm}(t)dt. \tag{3.77}$$

Apparently, the motions of a vehicle and a marker are independent from each other. By using the Jacobian matrix A shown in (3.19) used for a standard EKF, the Jacobian matrix for marker-aided EKF (A_m) can be expressed as follows:

$$A_m = \begin{bmatrix} A & 0_{15\times 9} \\ 0_{9\times 15} & \left[\begin{array}{c|c|c} & & \\ \hline & & I_3 \\ \hline & & \end{array}\right] \end{bmatrix}. \tag{3.78}$$

The process noise covariance matrix Q_m is much the same as Q as shown in (3.34):

$$Q_m = \begin{bmatrix} Q & \\ & 0_{9\times 9} \end{bmatrix}. \tag{3.79}$$

3.2.3 Update estimated states with vision

Firstly, let us express the output of the estimator y_m. The output of the estimator regarding the orientations is the attitude of the marker with respect to the body expressed in the axis-angle representation. The output regarding the position of the marker is the position of the

marker with respect to the body expressed in the body frame:

$$y_m = \begin{bmatrix} \hat{\rho}_m^b \\ \hat{p}_{bm}^b \end{bmatrix}. \tag{3.80}$$

By expressing the output of the EKF in body coordinate frame, the differentiation techniques explained in Subsection 2.3.2 can be utilized. Like the attitude of the body, the rotation of the marker is stored as a rotation matrix. The log map of the output is computed as follows:

$$\hat{\rho}_m^b = \log(\hat{R}_m^b) = \log(\hat{R}_b^{nT}\hat{R}_m^n). \tag{3.81}$$

For a marker detection, the output matrix is expressed as follows:

$$H_m = \left[\begin{array}{c|c|c|c|c|c|c} -I & & & & \hat{R}_m^b & & \\ \hline [\hat{p}_{bm}^b]_\times & -\hat{R}_b^{nT} & & & & \hat{R}_b^{nT} & \end{array}\right], \tag{3.82}$$

where

$$\hat{R}_m^b = \hat{R}_b^{nT}\hat{R}_m^n, \tag{3.83}$$

and

$$\hat{p}_{bm}^b = \hat{R}_b^{nT}(-\hat{p}_{nb}^n) + \hat{R}_b^{nT}(\hat{p}_{nm}^n) \tag{3.84}$$

$$= \hat{R}_b^{nT}(\hat{p}_{nm}^n - \hat{p}_{nb}^n) \tag{3.85}$$

$$= \hat{R}_b^{nT}\hat{p}_{bm}^n. \tag{3.86}$$

The non-trivial entries of the Jacobian matrix are derived as follows:

$$\frac{\partial y_{m,p}}{\partial \hat{\rho}_b^n} = \frac{\partial \hat{R}_b^{nT} \hat{p}_{bm}^n}{\partial \hat{\rho}_b^n} \ (\because (3.86)) \tag{3.87}$$

$$= [\hat{R}_b^{nT} \hat{\rho}_{bm}^n]_\times \ (\because (2.73)) \tag{3.88}$$

$$= [\hat{\rho}_{bm}^b]_\times, \tag{3.89}$$

$$\frac{\partial y_{m,p}}{\partial \hat{p}_{nb}^n} = \frac{\partial \hat{R}_b^{nT} (\hat{p}_{nm}^n - \hat{p}_{nb}^n)}{\partial \hat{p}_{nb}^n} \ (\because (3.85)) \tag{3.90}$$

$$= \frac{\partial \hat{R}_b^{nT} (-\hat{p}_{nb}^n)}{\partial \hat{p}_{nb}^n} \tag{3.91}$$

$$= -\hat{R}_b^{nT}, \tag{3.92}$$

$$\frac{\partial y_{m,p}}{\partial \hat{p}_{nm}^n} = \frac{\partial \hat{R}_b^{nT} (\hat{p}_{nm}^n - \hat{p}_{nb}^n)}{\partial \hat{p}_{nm}^n} \ (\because (3.85)) \tag{3.93}$$

$$= \frac{\partial \hat{R}_b^{nT} (\hat{p}_{nm}^n)}{\partial \hat{p}_{nm}^n} \tag{3.94}$$

$$= \hat{R}_b^{nT}. \tag{3.95}$$

For orientation parts, the derivative evaluated around a body frame is computed using (2.80). Note the example is specific to ρ_n^b, but the Lie algebra in this section ρ_b^n is defined in an opposite direction. Therefore, the final outcome also becomes opposite. The derivative around a marker frame is derived using (2.79), where f is simply a zero rotation around the marker frame. Here, it is useful to revisit (3.72), which is a measurement error covariance matrix expressed in a camera coordinate. As is explained in this subsection, the output of the filter is expressed in the body frame. However, only the covariance of the detection algorithm in the camera coordinate is known. Thus, the covariance matrix used

in Kalman equation is converted as follows:

$$\Sigma_m^c = G\Sigma_m^c G^T, \tag{3.96}$$

where

$$G = \begin{bmatrix} R_c^b & \\ & R_c^b \end{bmatrix}. \tag{3.97}$$

The notation follows the standard state space expression, which is

$$x_{k+1} = Fx_k + Gu_k, \tag{3.98}$$

where G is a discrete-time input matrix and u_k is a input vector. In this system, the input is a measurement noise. Finally, the attitude residuals are expressed as follows:

$$\begin{aligned} \epsilon_{m\rho} &= \log(\exp([\tilde{\rho}_m^b]_\times)(\exp([\hat{\rho}_m^b]_\times))^{-1}) \\ &= \log(\exp([\tilde{\rho}_m^b]_\times)(\exp([\hat{\rho}_m^b]_\times))^T) \\ &= \log(R_b^{cT}\tilde{R}_m^c(\hat{R}_b^{nT}\hat{R}_m^n)^T), \end{aligned} \tag{3.99}$$

and the residual of marker positions are as follows:

$$\begin{aligned} \epsilon_{mp} &= \tilde{p}_{bm}^b - \hat{p}_{bm}^b \\ &= (p_{bc}^b + R_b^{cT}\tilde{p}_{cm}^c) - \hat{R}_b^{nT}(\hat{p}_{nm}^n - \hat{p}_{nb}^n). \end{aligned} \tag{3.100}$$

Since vision produces multiple measurements of a marker, the update phase needs to find the most appropriate measurement. Also, there is a possibility of multiple markers being in the line of sight, and each EKF needs to be associated with the measurement that is created by a tracked target. Here, the Mahalanobis distance of measurement residuals is used to choose and associate a measurement with each EKF. Note that the covariance of

measurement residuals can be computed through the estimated error covariance of EKF as follows:

$$\begin{aligned} E[\epsilon_k \epsilon_k^T] &= E[(z_k - H_k \hat{x}_k^-)(z_k - H_k \hat{x}_k^-)^T] \\ &= E[\{H_k(x_k - \hat{x}_k^-) + \sigma\nu_k\}\{H_k(x_k - \hat{x}_k^-)^T + \sigma\nu_k\}^T] \\ &= H_k E[e_k^- e_k^{-T}] H_k^T + E[\sigma_k \sigma_k^T] \; (\because e_k^- = x_k - \hat{x}_k^-) \\ &= H_k P_k^- H_k^T + \Sigma_k = S. \end{aligned} \tag{3.101}$$

The covariance of marker residuals (S_m) is computed by replacing Σ_k with Σ_m^c and P_k^- with a block matrix of error covariance matrix corresponding to the marker states. Therefore, the Mahalanobis (Z) distance of marker residuals is:

$$Z^2 = \epsilon_m^T S_m \epsilon_m, \tag{3.102}$$

where

$$\epsilon_m = \begin{bmatrix} \epsilon_{m\rho} & \epsilon_{mp} \end{bmatrix}^T. \tag{3.103}$$

A constant threshold regarding the Mahalanobis distance is used to associate measurements with an EKF. This way, although the threshold for the Mahalanobis distance is constant, the threshold for the measurement association becomes adaptive. When an estimator is confident, it only uses a measurement close to a current estimation, but when an estimator is not confident, the criterion is loosened. With certain probability, this measurement phase is skipped to take false negatives into consideration.

3.2.4 Initialization

When initializing a marker-aided EKF, the states of a marker is initialized with the measurement available at that moment (denoted $\hat{R}^c_{m,\text{init}}$ and $\hat{p}^c_{cm,\text{init}}$).

$$\hat{R}^n_{m,\text{init}} = \hat{R}^n_{b,\text{init}} R^{cT}_b \hat{R}^c_{m,init} \tag{3.104}$$

and

$$\hat{p}^n_{nm,\text{init}} = \hat{p}^n_{nb,\text{init}} + \hat{R}^n_{b,\text{init}} (p^b_{bc} + R^{cT}_b \hat{p}^c_{cm,\text{init}}). \tag{3.105}$$

The states of the vehicle are initialized as described in Subsection 3.1.4. That creates $\hat{p}^n_{nb,\text{init}}$ and $\hat{R}^n_{b,\text{init}}$. When a marker state is initialized while an estimator has been running, simply the current estimation of the body states replaces them.

3.2.5 Merit of simultaneous estimation of vehicle and target states

The estimation of the target states can be built on top of the estimation of the vehicle states, in which the state estimation of the target does not have an influence of the estimation of the vehicle states. This approach has pros and cons. First, the dimensionality of the P matrix is reduced. This not only reduces the computational cost (note the cost of the propagation is $O(n^3)$ when n is the dimension of the state vector) but also improves the numerical stability. In addition, theoretically, the estimation errors decrease when any arbitrary additional sensor measurements are taken into account. When it comes to vision, the greatest benefit of simultaneous estimation is that this integration provides additional absolute position measurements. The vision measurements are only relative from a camera to a marker. However, while GPS is healthily working, the position of a marker is mapped in NED frame. By combining the absolute position of a marker and relative position from a camera, vision provides a position of vehicle. This effect is most remarkable when GPS systems are temporarily not available. Figure 3.2 shows the results of the simulation uti-

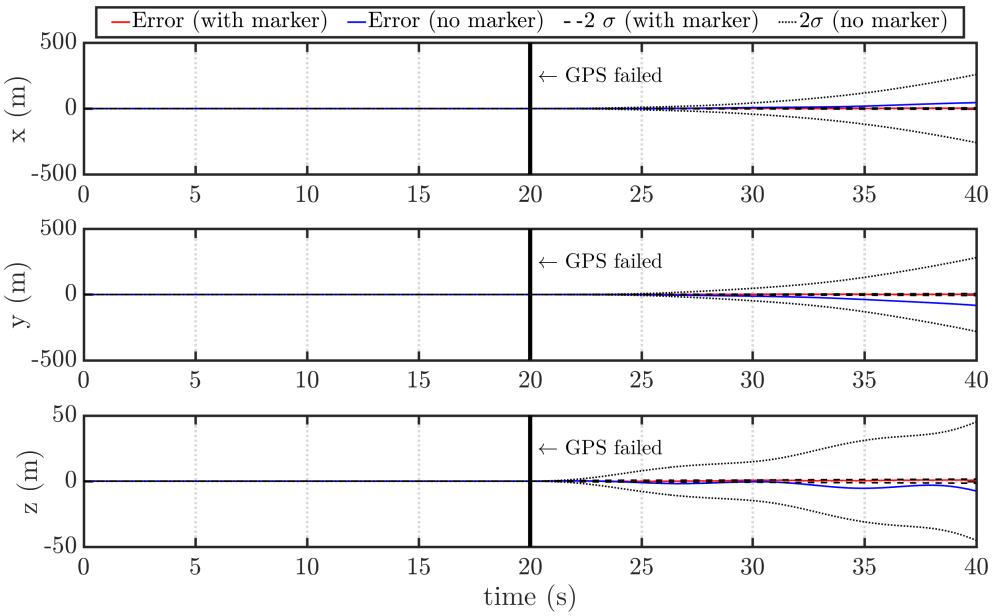
lizing the marker-aided EKF described in this section. In this simulation, GPS sensor fails in 20 seconds. Both the regular EKF and marker-aided EKF have a diverging covariance because they do not have absolute position measurements. However, having vision measurements integrated in the filter, the errors and covariance diverge by a far slower rate than the regular EKF. The numerical stability of the two EKF designs can be analyzed by looking at the condition number of the P matrix. Here, the condition number is computed as follows:

$$\kappa(P) = \frac{\sigma_{\max(P)}}{\sigma_{\min(P)}}, \tag{3.106}$$

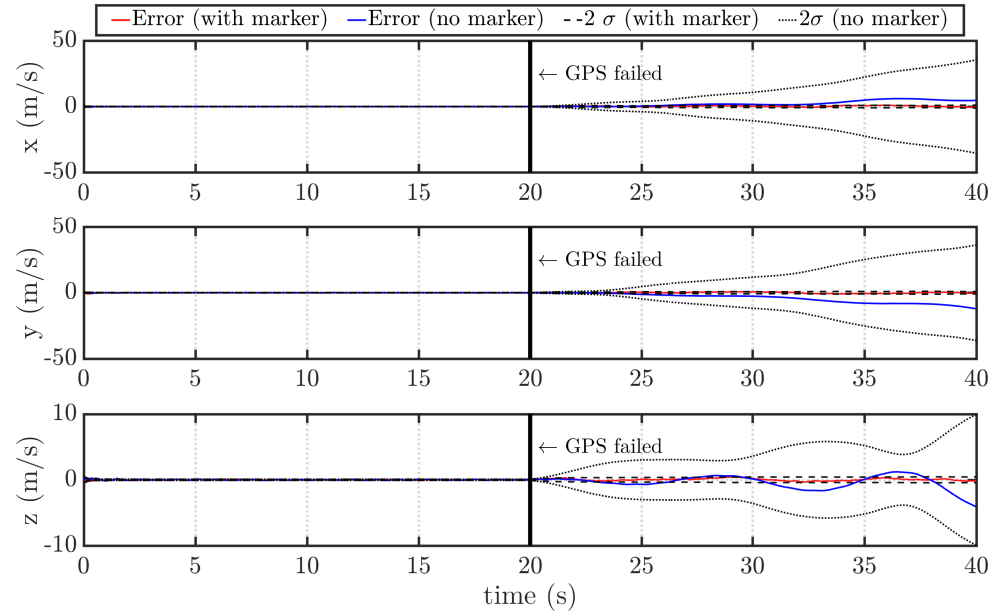
where $\sigma(P)$ is a singular value of the P matrix. Figure 3.3 shows the time history of the condition number in the same numerical simulation as Figure 3.2. For the marker-aided EKF, the states contributing the maximum singular value is the estimated position of the marker. Since the regular EKF does not include the marker states in the state vector, it has a lower condition number than the marker-aided EKF in usual circumstances. However, once the GPS signals are lost, the uncertainty of the vehicle position quickly increases. In this case, the marker-aided EKF is also favorable in a sense of numerical stability.

3.3 Combine particle filter with EKF

This section introduces extended Kalman particle filter (PF-EKF), which is also known as a bank of Kalman filter. In PF-EKF, each particle is an EKF explained in the previous section, but the particle filter technique is combined to acclimate to nonlinearity and multi-modility of the system. Particle filters (PFs) are methods for building a Monte Carlo approximation to the solutions of the Bayesian filtering equations. The core algorithms of particle filters are sequential importance sampling and sequential importance resampling [92], [93], or simply resampling. The convergence of the particle filter approximation is guaranteed by the central limit theorem [94]. Monte Carlo samples (denoted N) determines the error term $O(N^{-1/2})$. Particle filters have become a popular method for target



(a) Position



(b) Velocity

Figure 3.2: Figures show the behaviors of the estimation errors and their estimated covariance after GPS failed.

tracking because of their capability to estimate nonlinear/non-Gaussian states [95] and to track multiple targets [96]. For vision-based tracking, Cho, et al. [59] utilized the Rao-Blackwellized PF to track multiple bicycle using measurements from part-based HOG and

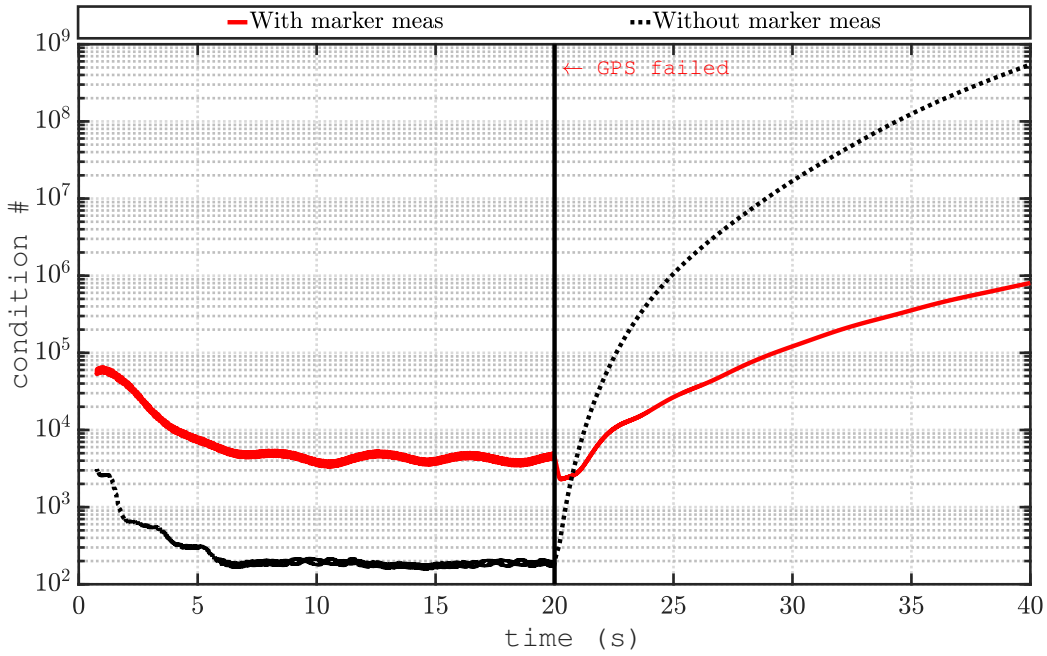


Figure 3.3: Figure shows the condition number of P in 15-state EKF (regular) and 24-state EKF (marker-aided) when a GPS failed in 20 s.

latent SVM [30]. Particle filters are theoretically superior to nearly all other model-based estimation techniques, such as extended Kalman filter (EKF) and unscented Kalman filter (UKF), but that is not necessarily the case in practice because of the 'curse' of the dimensionality [97], [98]; therefore, many algorithms are derived from particle filters to reduce computational cost. The most common approach is to form the importance distribution with a smaller Monte Carlo samples. Doucet, et al. [99] used Rao-Blackwellized particle filter (RBPF), which utilizes Monte Carlo approximation for only part of the estimated states and Van der Merwe, et al. [100] used the unscented particle filter to combine UKF and PF. This thesis suggests using the combination of an EKF and PF (PF-EKF) in order to estimate a multi-modal, nonlinear, and non-Gaussian model. This approach allows the filter to reduce the computational cost compared to a plain PF. Previous work in this field [101] has demonstrated that PFs may be used to estimate target states using a separated EKF that estimates vehicle states; however, this thesis estimates both vehicle and target states in the same filter in a SLAM paradigm. Another way to combine EKF and PF, RBPF, is discussed

in the next section.

3.3.1 Propagate, update, and resample

Each particle of the PF-EKF represents a state vector shown in (3.69), and it is propagated and updated as described in the previous section. One of the greatest advantages of this method over a multi-target tracking with a single filter is scalability. The most expensive operation of an EKF is a propagation of an error covariance matrix, which is $O(N_x^3)$, where N_x is the dimension of state vector. Since each target adds nine dimensions to the state vector (three-dimensional orientation, position, and velocity), tracking N_{target} targets costs $O((15 + 9N_{\text{target}})^3)$. On the other hand, in this PF-EKF, the dimension of the state vector is limited to 24. When using N_{particle} particles, it costs $O(24^3 N_{\text{particle}})$. In the update phase of EKF, a likelihood (q) parameter (a.k.a. conditional relative likelihood or simply called weight) of each particle is computed based on the measurement residual. Note a multivariate normal distribution is expressed as:

$$p_X(x) = \frac{\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)}{\sqrt{(2\pi)^k \det(\Sigma)}}. \tag{3.107}$$

The likelihood parameter is expressed as follows [79]:

$$q \sim \frac{1}{(2\pi)^{m/2}\det(\Sigma_m^c)^{1/2}} \left(\exp\left(\frac{-(z-\hat{z})^T\Sigma_m^{c-1}(z-\hat{z})}{2}\right)\right), \tag{3.108}$$

where Σ_m^{c-1} is a measurement covariance matrix of marker measurements expressed in a camera coordinate, $z = \begin{bmatrix} \tilde{\rho}_{cm} & \tilde{p}_{cm}^c \end{bmatrix}^T$, and $\hat{z} = h(\hat{x}^-) = \begin{bmatrix} \hat{\rho}_{cm} & \hat{p}_{cm}^c \end{bmatrix}^T$. m is the dimension of measurements, which are six (three-dimensional position and orientation) in this case. The symbol $\sim$ means that the likelihood parameter is proportional to the right side of the expression (3.108). Since a relative likelihood is not changed by the constants, the actual

implementation computes the simplified one as follows:

$$q = \exp\left(-\frac{1}{2}(z-\hat{z})^T \Sigma_m^{c-1}(z-\hat{z})\right), \tag{3.109}$$

and each likelihood is normalized as follows:

$$q_i = \frac{q_i}{\sum_{j=1}^{N} q_j}, \tag{3.110}$$

where q_i is the likelihood of i-th particle. This parameter allows the evaluation of the pdf $p(z_k|x_{k,i}^-)$. Resampling is conducted based on this likelihood as shown in Algorithm 1. This algorithm is called low variance sampling [102]. Resampling is a method to remove particles with low weights and duplicate the ones with high weights. This operation is not usually performed every time particles are updated; instead, it is performed on every n steps. This n can be predefined or adaptively determined. One way to adaptively determine when to resample is to use the variance of the particle weights. Let N denote the total number of particles used in the filter [103]. Out of N particles, only part of the particles satisfying $N_{\text{eff}} <= N$ is effectively representing the states, where N_{eff} is expressed as following:

$$N_{\text{eff}} \approx \frac{1}{\sum_{i=1}^{N} (q^{(i)})^2}, \tag{3.111}$$

where $q^{(i)}$ is the normalized weight of particle i. Resampling can be performed by using this N_{eff}. For example, certain ratio of particles are resampled when $N_{\text{eff}} < N/10$ [104]. Particles with low likelihood are initialized near a measurement location or a confident particle.

3.3.2 Computing outputs

Since the output of our target tracker is directly used for closed-loop waypoint navigation, the output should be fed into the planner only when the estimator has met certain perfor-

Algorithm 1 This algorithm is used to choose N new particles from old particles x in importance sampling and resampling with existing particle. The probability of a particle x_i being chosen for sampling is proportional to the associated weight q_i. This algorithm requires $O(N)$ time.

$x_{\text{sampled}} = \emptyset$
$r = U(0, N^{-1})$ ▷ $U(a, b)$ is a uniform random number between a and b
$c = q_1$
$i = 1$
for $n = 1$ to N **do**
  $U = r + (n-1) \cdot N^1$
  while $U > c$ **do**
    $i = i + 1$
    $c = c + q_i$
  end while
  add x_i to x_{sampled}
end for
return x_{sampled}

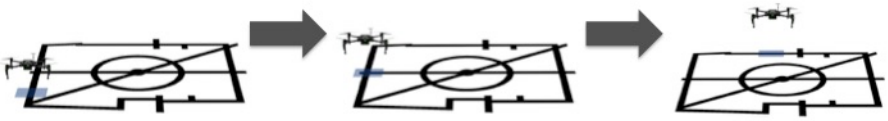
mance criteria. In the system described, the performance is evaluated using the covariance of the particles. Unlike the Kalman filter or other unimodal-based filters, the covariance of all particles is not the best tool to evaluate the performance. Instead of directly using the P of each particle, part of the particles are chosen based on their estimated error covariance (P) and likelihood. Then, the weighted average based on their likelihood parameter is used as the navigation solution. Computing the mean in vector space is straightforward, but the attitude of the vehicle and the marker are stored as a rotation matrix. Here, a rotation mean is computed by using Algorithm 2. A single rotation matrix is computed from many estimates, and this is called a 'single rotation averaging' [105]. Angular distance of two matrices $S, R \in SO(3)$ is expressed as follows:

$$d(S, R) = ||\log(\mathrm{SR}^{\mathrm{T}})||_2. \tag{3.112}$$

The mean of rotation matrices is computed to minimize the sum of the angular distance $C(R) = \sum_{i=1}^{N} d(S, R)^2$. This mean is known as Karcher mean of the rotations. The convergence of this algorithm is proven in [106] by Manton.

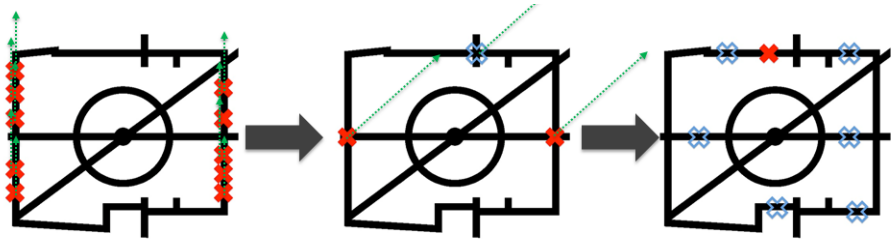
Algorithm 2 This algorithm compute the geodesic L_2 mean R_m on SO(3) from N rotation matrices.

$R_m \leftarrow R_1$
while true **do**
  $r = \frac{1}{n}\sum_{i=1}^{N} \log(R_m^T R_i)$
  if $||r|| < \epsilon$ **then** ▷ Choose a tolerance $\epsilon > 0$
    return R_m
  end if
  $R_m \leftarrow R_m \exp([\mathrm{r}]_\times)$
end while



(a) Vehicle motion

(b) On-board images



(c) Particle allocation

Figure 3.4: Figure (a) shows vehicle locations, and (b) and (c) are the corresponding on-board images and possible particle locations. When each onboard image is processed as a single static measurement, all the blue marker locations are the possible vehicle locations. However, by combining the known vehicle dynamics, the particle locations (red makers) can be constrained to only the ones that match with the vehicle motion.

3.3.3 Merit of particle filter

Utilizing a particle filter for target tracking enables further benefits from a single-hypothesis approach. The most important point of the use of the particle filter is the ability to maintain multiple hypotheses for the location of the target in the presence of ambiguous measurements, and to refine those hypotheses as vehicle motion and additional measurements. In this way, the target detection is dramatically more successful and is able to track targets that Kalman filter-based approaches cannot. The success of the particle filter is demonstrated with an example shown in Figure 3.4, where (a) is vehicle locations and (b) and (c) are the corresponding on-board images and particle locations. The vehicle is attempting to land from the corner of the target as shown in the left of the Figure 3.4-(a). When only a vertical line is seen, all the vertical edges display a high matching score; thus, the particle locations, denoted as red markers, are distributed all over the vertical edges. Then, the vehicle moves as shown in Figure 3.4-(a). When processing an on-board image as a single static measurement, all the blue markers are possible locations. However, a particle that does not match with the vehicle motion has a low likelihood q (as shown in (3.109)), and will not be resampled. With the particle filter, it can uniquely identify the target location when very little information is available from an on-board image, which is especially powerful for vision-based landing.

3.4 Rao-Blackwellized particle filter

The last estimation algorithm introduced in this section is Rao-Blackwellized particle filter (RBPF). Rao-Blackwellization [107], [108], [109] is to use the following conditioning inequality for an estimator:

$$\mathrm{var}(E(\delta(X)|Y)) \leq \mathrm{var}(\delta(X)). \tag{3.113}$$

The name comes from the theorem this inequality is associated with, the Rao-Blackwell theorem [110], [111]. RBPF [99], [95], [112], also known as marginalized particle filter [113] and mixture Kalman filter [114], is the particle filter using this conditioning inequality. By evaluating some of the filtering equation in a framework of analytical filters such as EKF and UKF instead of computing everything through Monte Carlo approach, it is often possible to reduce the variance of the estimation. Estimating the entire state vector in the framework of naive particle filter is practically impossible for the work presented here due to the dimensionality of the state vector and the necessity to run in real-time. The PF-EKF approach uses the optimal measurement update based on the linearized state space through Kalman equation and uses the PF framework only for resampling and computing the final estimated values. However, the vehicle states and sensor biases are estimated well through an EKF, and most of the nonlinearity and non-Gaussianity come through the visual measurements of a marker. The PF-EKF approach allows a multiple-hypothesis estimation, but each particle maintains an EKF structure; therefore, it does not fully enable a nonlinear estimation. Plus, each particle needs to have the full size of the error covariance matrix, and propagation of the error covariance matrix is computationally a huge burden. In RBPF, the state vector is partitioned into two parts: EKF and PF states. This allows the filter to reduce the dimensionality of the covariance matrix P, which is preferable in the sense of computational power and numerical stability.

3.4.1 Derivation of RBPF for vision-aided navigation

The RBPF in this work partitions the estimated states as follows:

$$\hat{x} = \left[\underbrace{\left[\hat{\rho}_b^n, \hat{p}_{nb}^n, \hat{v}_{nb}^n, \hat{b}_{\text{gyro}}, \hat{b}_{\text{acc}}, \hat{v}_{nm}^n \right]}_{\text{EKF states}} \quad \underbrace{\left[\hat{\rho}_m^n, \hat{p}_{nm}^n \right]}_{\text{PF states}} \right] \in \mathbb{R}^{24}. \tag{3.114}$$

The same notations from the previous section are used, and the combined state vector is exactly the same as the ones used in EKF and PF-EKF. For the rest of this section, the

following simplified notations are used for the EKF and PF states:

$$x^{kf} = \begin{bmatrix} \rho_b^n & p_{nb}^n & v_{nb}^n & b_{\text{gyro}} & b_{\text{acc}} & v_{nm}^n \end{bmatrix} \in \mathbb{R}^{18}, \tag{3.115}$$

and

$$x^{pf} = \begin{bmatrix} \rho_m^n & p_{nm}^n \end{bmatrix} \in \mathbb{R}^{6}. \tag{3.116}$$

Note that the velocity of the marker $\hat{v}_{nm}^n$ is moved to the left so that the first part becomes the EKF states, and the rest becomes the PF states. The prediction of the states are conducted in the same manner described in the previous section; however, the estimated error covariance matrix no longer contains the elements for marker states ($P \in \mathbb{R}^{18\times 18}$). The goal of this RBPF is to estimate the density of the states and latent variable, u. From (2.118),

$$p(x_k, u_k|z_{1:k}) = p(x_k^{kf}, x_k^{pf}, u_k|z_{1:k}) = p(x_k^{kf}|x_k^{pf}, u_k, z_{1:k})p(x^{pf}, u_k|z_{1:k}), \tag{3.117}$$

where

$$u_k = z_{pf,k} = p_{nm,k+1}^n - p_{nm,k}^n. \tag{3.118}$$

This is called the pseudo measurement, or latent variable. The k is the time index. All the measurements except for vision solely update the vehicle states and sensor biases, and they are used in a standard EKF fashion. Let x^{kf-} denote *a priori* states, which are the states conditioned by all other sensors up to current time step except the vision measurement. The density of *a priori* states is expressed as follows:

$$p(x_k^{kf-}|x_{k-1}^{pf}, u_{k-1}, z_{m,1:k-1}, z_{gps,1:k}, z_{mag,1:k}, z_{baro,1:k}, \ldots). \tag{3.119}$$

Note z includes the measurements of all sensors, such as GPS (z_{gps}), magnetometer (z_{mag}), barometric pressure sensor (z_{baro}), etc. Therefore, conditioning the density of x_k^{pf} by *a*

priori KF states are equivalent to the original problem:

$$p(x_k^{pf}, u_k | z_{1:k}) = p(x_k^{pf}, u_k | z_{1:k}, x_k^{kf-}). \tag{3.120}$$

In the prediction phase of particle filter, the estimated position of the marker is propagated as follows:

$$x^{pf}_{p^n_{nm},k+1} = x^{pf}_{p^n_{nm},k} + x^{kf}_{v^n_{nm},k} dt. \tag{3.121}$$

Measurement residuals are also computed in a similar fashion:

$$\epsilon_{m,rbpf} = \epsilon(x_{k-1}^{pf}, x_k^{kf-}), \tag{3.122}$$

where x^{pf} and x^{kf} are used to satisfy (3.99) and (3.100). Then, the output of the particle filter is used to update the estimated EKF states and their covariance matrix:

$$\hat{x}^{kf+} = \hat{x}^{kf-} + L(z_{pf} - F_{pf}\hat{x}^{kf-}), \tag{3.123}$$

where

$$L = F_{kf} P^+ F_{pf}^T N^{-1} \in \mathbb{R}^{18\times 3}, \tag{3.124}$$

$$N = F_{pf} P^+ F_{pf}^T + Q_{v^n_{nm}} \in \mathbb{R}^{3\times 3}. \tag{3.125}$$

Here, the velocity of the marker is estimated using the first order approximation as follows:

$$\hat{v}^n_{nm,k} = \frac{p^n_{nm,k+1} - p^n_{nm,k}}{dt} = \frac{z_{pf,k}}{dt}. \tag{3.126}$$

Therefore,

$$F_{pf} = \left[\begin{array}{c|c} 0_{3\times 15} & dt I_3 \end{array} \right]. \tag{3.127}$$

The F_{kf} is the matrix expressed as follows:

$$F_{kf} = e^{A_{kf}dt}, \tag{3.128}$$

where

$$A_{kf} = \frac{\partial f(x)}{\partial \hat{x}^{kf}} \in \mathbb{R}^{18\times 18}. \tag{3.129}$$

Using the A matrix (3.19) derived in Section 3.1,

$$F_{kf} = \left[\begin{array}{c|c} I + Adt & \\ \hline & I \end{array}\right], \tag{3.130}$$

and the estimated error covariance is updated as follows:

$$P^{+} = P^{-} - LNL^{T} \in \mathbb{R}^{18\times 18}. \tag{3.131}$$

This is the way x^{kf} is updated with z_{pf}. The vehicle states are also updated with the measurement of the marker position and orientation. Since the marker position and orientation are excluded from the EKF state, H matrix is shortened compared to (3.82) as follows:

$$H_{m,rbpf} = \left[\begin{array}{c|c|c|c|c|c} -I & & & & & \\ \hline [\hat{p}^{b}_{bm}]_{\times} & -\hat{R}^{nT}_{b} & & & & \end{array}\right]. \tag{3.132}$$

3.4.2 Initialization of velocity

When a particle is instantiated based on a measurement, initial velocities of some particles are estimated through sequential raw measurements. For each particle, measured marker positions are computed as follows:

$$\tilde{p}^{n}_{nm} = \hat{p}^{n}_{nb,i} + \hat{R}^{n}_{b,i}(p^{b}_{bc} + R^{cT}_{b}\tilde{p}^{c}_{cm}). \tag{3.133}$$

The initial velocity, therefore, is computed as follows:

$$\hat{v}^n_{nm,\text{init}} = \frac{\tilde{p}^n_{nm}(k+1) - \tilde{p}^n_{nm}(k)}{dt_m}, \tag{3.134}$$

where dt_m is the time difference between the current and last marker measurements. However, this method requires correct pairs of measurement associated across different image frames. Since each image produces multiple measurements, this operation is not straight forward. Equation (3.133) is computed for every measurements, and they are saved. When a new measurement is available, (3.133) is computed and compared with the last measurements. The pairs that have the smallest Euclidean distance are used to compute the initial velocity. Another way to initialize the velocity of the particle is to give a small random velocity. Through sequential importance sampling, only the particles with accurate velocity survive; however, this method takes more time to converge. Also, the magnitude of the initial random velocity significantly affects the performance of the estimator.

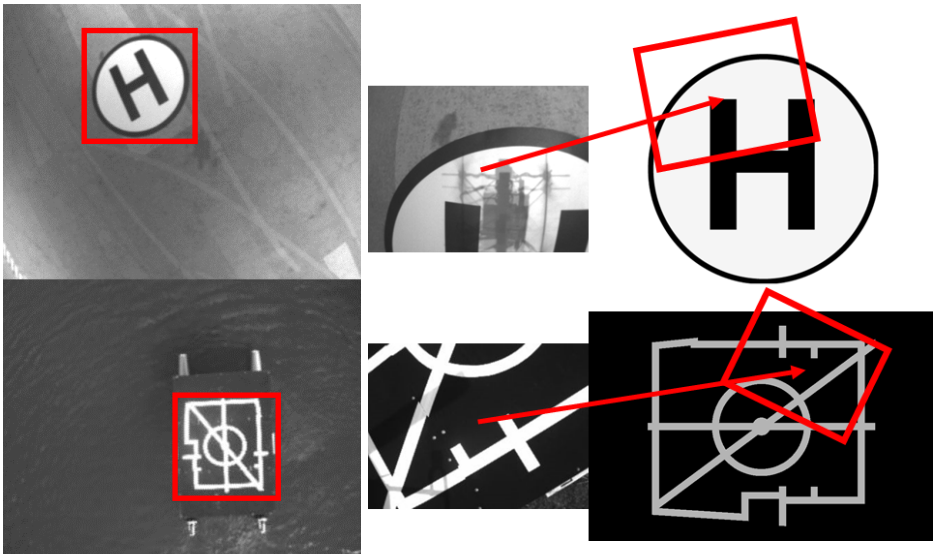
CHAPTER 4
DETECTION ALGORITHMS FOR LANDING

This chapter describes a newly developed detection framework used to observe a landing site. The framework is used when a custom-prepared landmark cannot be used, and the vision system is forced to observe an arbitrary landmark. Examples of this case in realistic conditions would be an existing navy deck pattern or a standard H-shaped helipad. This framework detects the center of the landing site from a partial observation of the landmark so that the system can observe a landing site even when a camera is very close to a target. This chapter introduces the implementation of the algorithm and the way to integrate these visual measurements in a visual SLAM paradigm.

4.1 Detecting a partial marker

A unique capability of this detection algorithm is to be able to locate an observed partial landmark with respect to the known full landmark (Figure 4.1). A standard detection task is to make a decision at every point of an input image whether or not there is a target feature at that point. Figure 4.1a shows an example, where the input image is an on-board camera image, and the target is a standard H-shaped helipad and a navy deck pattern. The rectangles represent the locations of the features recognized as a target. Detecting a distinct marker when an entire marker is in view is a well-studied area. Indeed, Nakamura, et al. demonstrated that a machine-learning-based algorithm can robustly detect a navy deck pattern using the Haar-like feature detector from an on-board camera of a flying UAV [25]. However, when a vehicle descends to land, an on-board camera is very close to a landing landmark, and the entire landmark is not in view. The detection algorithm used by Nakamura, et al. is still able to recognize this as a part of a landmark and locate the observed location with respect to a known full landmark as shown in Figure 4.1b. The new vision

system is added on top of the robust learning-based detection algorithms when the system can not observe a full landmark, which happens when a vehicle descends to land, a landing site makes an abrupt motion, and a disturbance does not allow a vehicle to fly directly over a target. This algorithm assumes that the system knows a marker and its scale.



(a) Standard detection task (b) Portion detection task

Figure 4.1: Standard detection vs detection required for landing

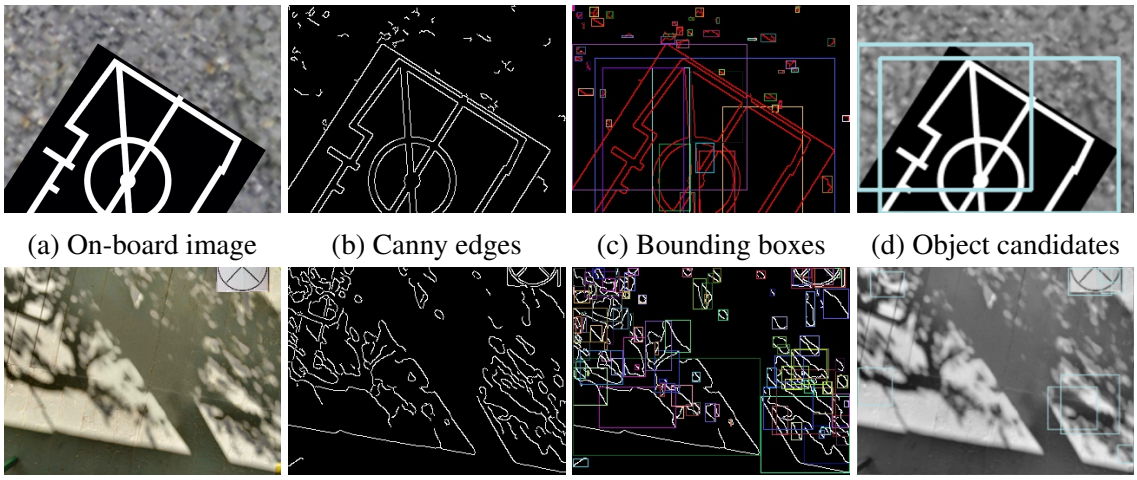
4.1.1 Extract object candidates

When an entire target is visible on an on-board image, the standard template matching technique and a state-of-the-art detection framework is a powerful tool; however, when the system does not observe an entire target, as is often the case with landing, especially at the final stage of landing, the system needs to extract object candidates before applying a template matching. There are many algorithms to extract object candidates [115], [116], and the main motivation of the development of these algorithms is to reduce the computational cost. Most of the detection algorithms, including the work presented in this section, can theoretically be applied to any part of the image and any combination of the window size, stride, etc. However, this approach is impractical in a sense of computational cost. Using an object candidate approach, the interesting regions of the image are chosen first,

and more complex detection algorithms are applied to these regions to detect and classify the regions. One caveat of this method is that the regions that are missed in the phase of the candidate extraction will never be evaluated in the following phases. Therefore, the candidate extraction is tuned be less conservative, and it includes many false positives at this point. In this work, contour trees [117] are used for candidate extraction. A contour is made up of feature points that represent a curve, and a box that bounds a curve is extracted as an object candidate. This vision system first extracts object candidates from an on-board camera image. Figure 4.2 explains the key steps to extract object candidates. The raw on-board image (Figure 4.2a) is first gray scaled, and the edges of the image are computed using the Canny edge detection algorithm as shown in Figure 4.2b. A contour tree is computed and the bounding boxes of each contour are calculated (Figure 4.2c). The 3-D positions of the vehicle and the marker are estimated, and the vision system computes the size of the marker that appears in an image based on the estimated position.. (The methods to estimate these positions are described in the next section.) Using this estimated size, the bounding boxes that do not match with the expected size are filtered out. The final object candidates (blue rectangles) are obtained after this filtering operation (Figure 4.2d).

4.1.2 Template matching with sliding window

The object candidates extracted with the method explained in the previous subsection are used for detection. The bounding boxes that contain the object candidates are swept across an image that contains the known marker, and the matching score between the object candidate and the marker is computed. The known marker image is scaled by using the known physical dimensions of the target and the estimated distance between the marker and the on-board camera. The marker image is rotated using the estimated attitude of the marker. The system computes a normalized cross correlation (NCC) [118] of the object candidate



(a) On-board image (b) Canny edges (c) Bounding boxes (d) Object candidates

(e) On-board Image (f) Canny edges (g) Bounding boxes (h) Object candidates

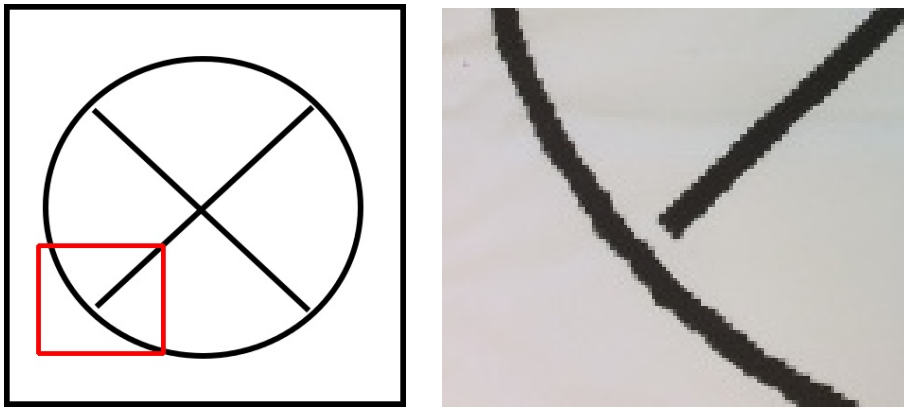
Figure 4.2: Figures explain the procedure to extract object candidates using two examples. The edges of the images are computed with the Canny edge detector, and the contours are computed by linking up the edges (b, f). All possible bounding boxes of all contours are computed (c, g). Only the contour that satisfies the dimension test is used as a object (d, h).

and marker images as a matching score as shown below:

$$\gamma(u,v) = \frac{\sum_{x,y} |f(x,y) - \bar{f}_{x,y}|(t(x-u,y-v) - \bar{t})}{\sqrt{\sum_{x,y} |f(x,y) - \bar{f}_{x,y}|^2 (t(x-u,y-v) - \bar{t})^2}}, \tag{4.1}$$

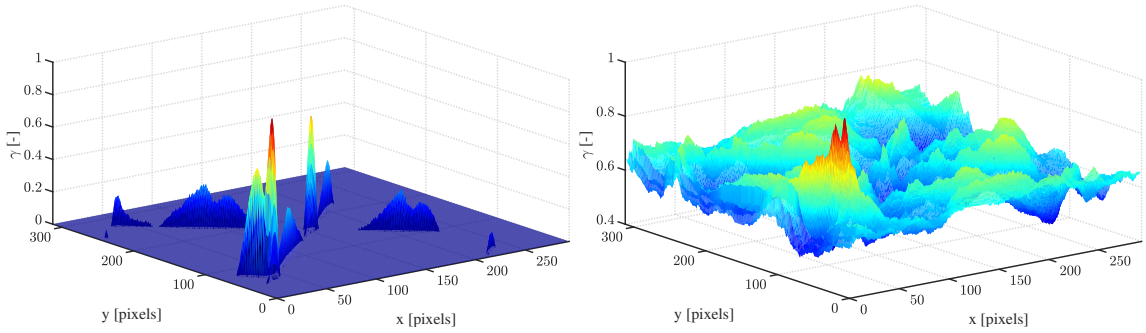
where (u,v) is the location on a source image (f), and t denotes a template image. The bar represents the average intensity of the image; thus, $\bar{f}$ and $\bar{t}$ are the average intensity of the source and the template images, respectively. The popular feature vectors such as scale invariant feature transform (SIFT) and histogram of oriented gradients (HoG) [29] are also tested in this detection algorithm. However, because of the following two reasons, the NCC is chosen for a baseline matching score. The first reason is computational cost. In this algorithm, the feature vector of the entire bounding box needs to be computed when the vision system obtains a new on-board camera image. With a featureful background, there are many object candidates, and computing high dimension feature vectors such as SIFT and HoG was computationally expensive. The second reason is that NCC showed a sharper peak in the distribution of the matching score, and setting a threshold to decide

whether or not the part of an image contains a target feature is easier compared to the other feature vectors. Figure 4.4 shows an example of the matching score distributions.



(a) Known target (b) Simulated on-board image

Figure 4.3: Figures show a set of the images used in the test. A simulated on-board image is taken (b) and it is compared with the known template image (a) using a sliding window approach. The results of this test is shown in Figure 4.4.

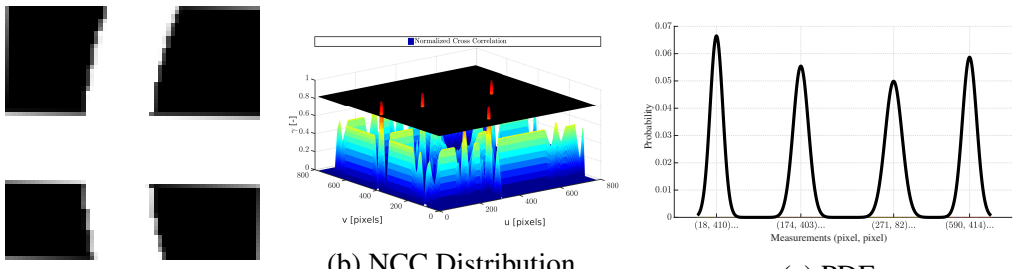


(a) NCC (b) HoG

Figure 4.4: Figures show the distribution of the matching score. 4.4a is the distribution of normalized cross correlation and 4.4b is the one of histogram of oriented gradient.

For a standard detection task, any location that has a higher matching than a threshold is recognized as a target. However, the vision algorithm presented here is inherently multi-modal and often not accurate enough with a single shot. Figure 4.5 shows an example. When an on-board camera sees a very small part of the landmark as shown in Figure 4.5a, it is extremely challenging to determine which part of the landmark is observed. In fact, as the result distribution indicates (Figure 4.5b), all the locations that have

vertical and horizontal crossings show a great matching score. The black plane shown in Figure 4.5c is the threshold applied to the matching score, and all the locations beyond this threshold are stored as detected target locations. These locations are weighted by their NCCs, and the final output of the vision algorithm given to the estimator is the probability distribution, which contains the possible target locations and the corresponding probability. Figure 4.6 shows the error distributions of the marker measurements using this portion detection. These results are based on the samples obtained from an image-in-the-loop simulation. 30000 samples of true and measured positions of the marker are recorded. The results show that the error distributions are highly non-Gaussian and multi-modal.



(a) partial marker

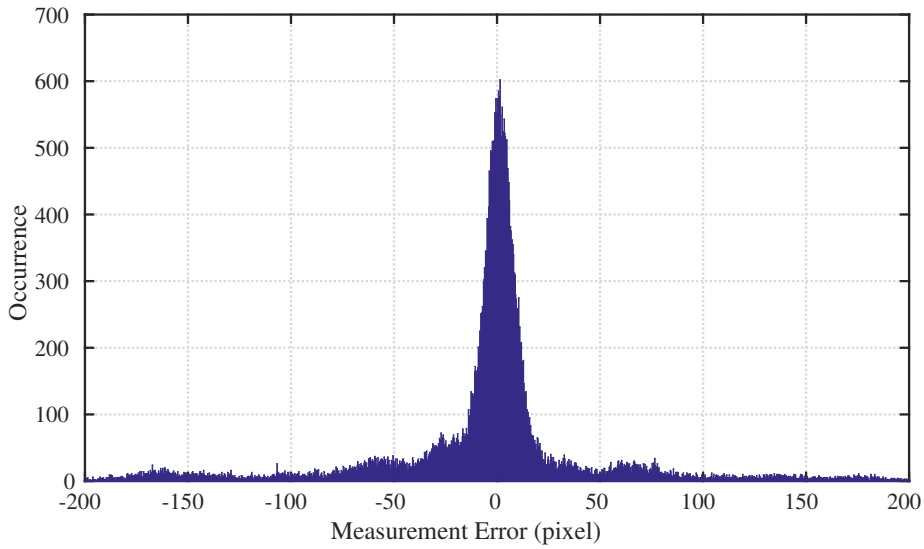
(b) NCC Distribution

(c) PDF

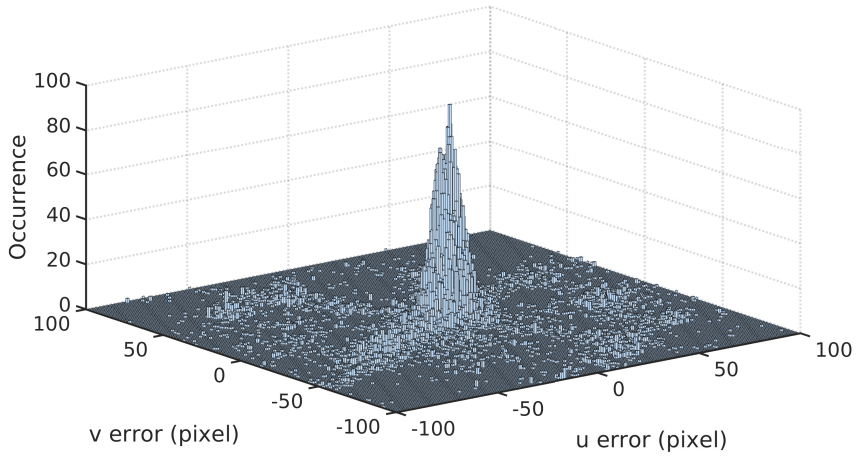
Figure 4.5: Figure 4.5a is an example of the on-board camera image, and 4.5b is an example of the multi-modal outputs. Instead of using a single point that has the highest matching score, all the locations beyond the threshold are stored with the corresponding matching score as shown in 4.5c.

4.2 Use of vision for state estimation

Vision system produces two different types of the outputs depending on the vehicle altitude. First,the critical altitude (z_{crit}) must be defined. The critical altitude is the altitude where the entire marker can not be in view. It is computed based on the estimated distance from the on-board camera to the marker, known physical dimensions of the marker, and the camera calibration parameters. While the vehicle is flying higher than this critical altitude, a marker is detected using the machine-learning-based algorithm trained in previous work [25]. The SIFT features of the detected area are computed, and the matches are generated using a



(a) 1-D error

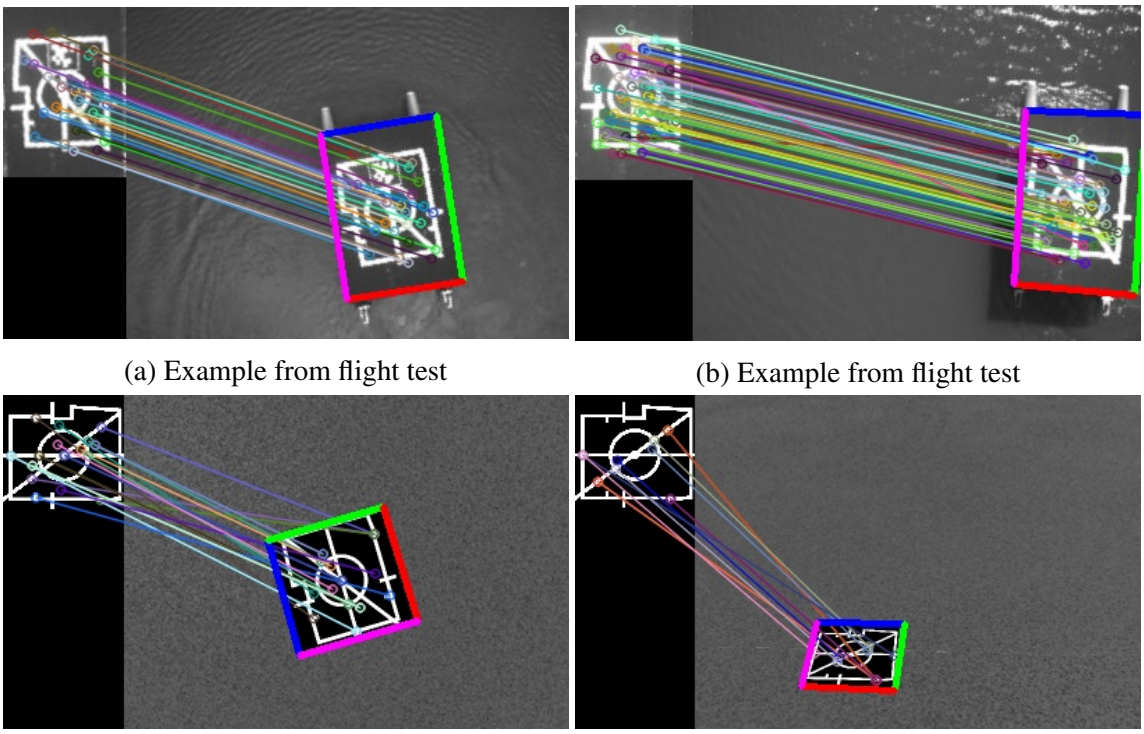


(b) 2-D

Figure 4.6: Figures show the measurement error distribution. Figure 4.6a shows the Euclidean distance error, and Fig 4.6b shows the distribution of error of $[u\ v]$.

FLANN-based algorithm [119], which utilizes an approximate nearest neighbor search. Figure 4.7 shows an example of the feature matching.

The knowledge of the physical dimensions of the marker allows the vision system to measure the three-dimensional translations and rotations of the marker with respect to the on-board camera which are denoted $\tilde{R}^c_m$ and $\tilde{p}^c_{cm}$. Note that the *tilde* symbol is used for a



(a) Example from flight test

(b) Example from flight test

(c) Example from simulation

(d) Example from simulation

Figure 4.7: Figures show examples of feature matching and pose extraction using a navy deck pattern. The SIFT features ([120]) are extracted from a known target and on-board image. The corresponding pairs are found using the RANSAC algorithm. The work here assumes that the vision system knows the physical dimensions of the target; thus, the translation and orientations of the target can be extracted.

measured value. Let $z_{\text{vision,high}}$ denote the measurements at altitude beyond z_{crit}. Then,

$$z_{\text{vision,high}} = \begin{bmatrix} \log(\tilde{R}^b_m) \\ \tilde{p}^b_{bm} \end{bmatrix} = \begin{bmatrix} \log(R^b_c \tilde{R}^c_m) \\ p^b_{bc} + R^b_c \tilde{p}^c_{cm} \end{bmatrix} \in \mathbb{R}^6. \tag{4.2}$$

The H matrix corresponding to this measurement is expressed as following:

$$H_{\text{vision,high}} = \left[\begin{array}{c|c|c|c|c|c|c} -I & & & & \hat{R}^b_m & & \\ \hline [\hat{p}^b_{bm}]_\times & -\hat{R}^{nT}_b & & & & \hat{R}^{nT}_b & \end{array}\right], \tag{4.3}$$

which is derived in (3.82). Once the vehicle descends lower than z_{crit}, the entire marker is not observed, and the measurement is the 2-D location of the marker expressed in pixels.

Let $z_{\text{vision,low}}$ denote the measurements at altitude below z_{crit}. Then,

$$z_{\text{vision,low}} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \frac{p^c_{cm,y}}{p^c_{cm,z}} + c_x \\ f_y \frac{-p^c_{cm,x}}{p^c_{cm,z}} + c_y \end{bmatrix} + \nu, \tag{4.4}$$

where f_x and f_y are the focal lengths of the on-board camera, c_x and c_y are the optical center in pixel, and ν is a measurement noise $\nu \sim N(0, R)$. This utilizes the pinhole camera model, which is explained in Figure 4.8.

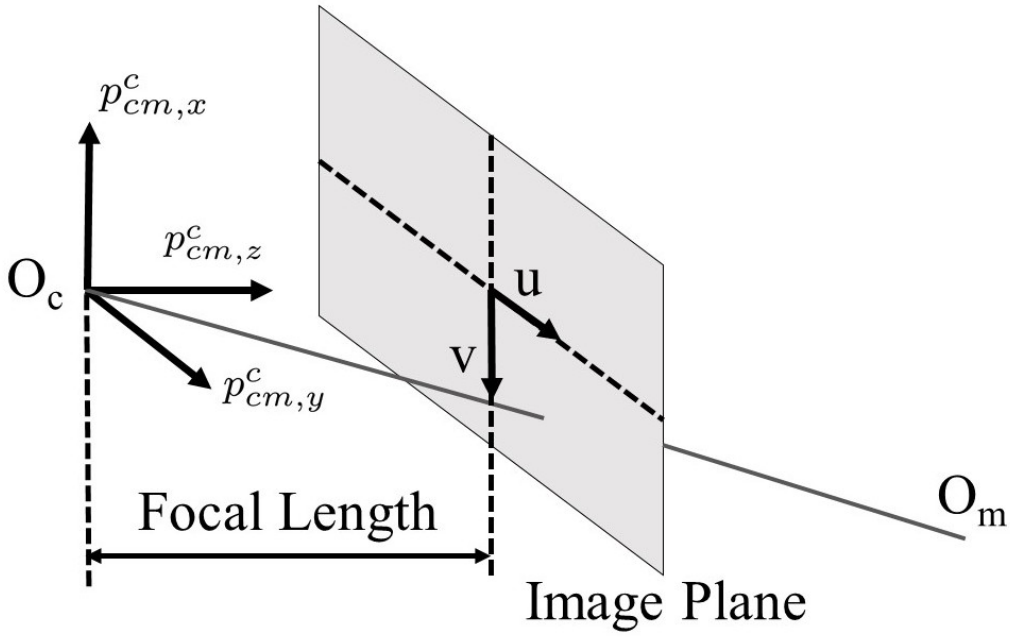


Figure 4.8: Figure explains the basic pinhole camera model. The position of the marker expressed in the world coordinate is projected to the image coordinate [u v].

The output $y_{\text{vision,low}}$ is computed in a similar fashion

$$y_{\text{vision,low}} = \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \begin{bmatrix} f_x \frac{\hat{p}^c_{cm,y}}{\hat{p}^c_{cm,z}} \\ f_y \frac{-\hat{p}^c_{cm,x}}{\hat{p}^c_{cm,z}} \end{bmatrix}. \tag{4.5}$$

The image-in-the-loop simulation uses this perfect pinhole camera; but, in flight tests, the perceived points $\begin{bmatrix} u & v \end{bmatrix}$ are expressed on an undistorted image using the calibration pa-

rameters. This 2-D measurement contributes to two elements of the $H_{\text{vision,low}}$:

$$\frac{\partial y_{\text{vision,low}}}{\partial \hat{p}^n_{nb}} = \frac{\partial z_{\text{vision,low}}}{\partial \hat{p}^c_{cm}} \frac{\partial \hat{p}^c_{cm}}{\partial \hat{p}^n_{nb}} \tag{4.6}$$

$$= \begin{bmatrix} \frac{\partial \hat{u}}{\partial \hat{p}^c_{cm,x}} & \frac{\partial \hat{u}}{\partial \hat{p}^c_{cm,y}} & \frac{\partial \hat{u}}{\partial \hat{p}^c_{cm,z}} \\ \frac{\partial \hat{v}}{\partial \hat{p}^c_{cm,x}} & \frac{\partial \hat{v}}{\partial \hat{p}^c_{cm,y}} & \frac{\partial \hat{v}}{\partial \hat{p}^c_{cm,z}} \end{bmatrix} \frac{\partial \hat{p}^c_{cm}}{\partial \hat{p}^n_{nb}} \; (\because (4.5)) \tag{4.7}$$

$$= \begin{bmatrix} 0 & f_x \frac{1}{\hat{p}^c_{cm,z}} & -f_x \frac{\hat{p}^c_{cm,y}}{\hat{p}^{c2}_{cm,z}} \\ -f_y \frac{1}{\hat{p}^c_{cm,z}} & 0 & f_y \frac{\hat{p}^c_{cm,x}}{\hat{p}^{c2}_{cm,z}} \end{bmatrix} (-\hat{R}^c_n), \tag{4.8}$$

and similarly,

$$\frac{\partial y_{\text{vision,low}}}{\partial \hat{p}^n_{nm}} = \frac{\partial z_{\text{vision,low}}}{\partial \hat{p}^c_{cm}} \frac{\partial \hat{p}^c_{cm}}{\partial \hat{p}^n_{nm}} \tag{4.9}$$

$$= \begin{bmatrix} 0 & f_x \frac{1}{\hat{p}^c_{cm,z}} & -f_x \frac{\hat{p}^c_{cm,y}}{\hat{p}^{c2}_{cm,z}} \\ -f_y \frac{1}{\hat{p}^c_{cm,z}} & 0 & f_y \frac{\hat{p}^c_{cm,x}}{\hat{p}^{c2}_{cm,z}} \end{bmatrix} \hat{R}^c_n, \tag{4.10}$$

,

$$\frac{\partial y_{\text{vision,low}}}{\partial \hat{\rho}^n_b} = \frac{\partial z_{\text{vision,low}}}{\partial \hat{p}^c_{cm}} \frac{\partial \hat{p}^c_{cm}}{\partial \hat{\rho}^n_b} \tag{4.11}$$

$$= \begin{bmatrix} 0 & f_x \frac{1}{\hat{p}^c_{cm,z}} & -f_x \frac{\hat{p}^c_{cm,y}}{\hat{p}^{c2}_{cm,z}} \\ -f_y \frac{1}{\hat{p}^c_{cm,z}} & 0 & f_y \frac{\hat{p}^c_{cm,x}}{\hat{p}^{c2}_{cm,z}} \end{bmatrix} R^c_b [\hat{p}^b_{bm}]_\times. \tag{4.12}$$

Note $\hat{p}^c_{cm} = -R^c_b p^b_{bc} + R^c_b \hat{R}^{nT}_b (\hat{p}^n_{nm} - \hat{p}^n_{nb})$; therefore,

$$\frac{\partial \hat{p}^c_{cm}}{\partial \hat{p}^n_{nb}} = \frac{\partial (-R^c_b p^b_{bc} + R^c_b \hat{R}^{nT}_b (\hat{p}^n_{nm} - \hat{p}^n_{nb}))}{\partial \hat{p}^n_{nb}} \tag{4.13}$$

$$= \frac{\partial R^c_b \hat{R}^{nT}_b (\hat{p}^n_{nm} - \hat{p}^n_{nb})}{\partial \hat{p}^n_{nb}} \tag{4.14}$$

$$= \frac{\partial R^c_b \hat{R}^{nT}_b (-\hat{p}^n_{nb})}{\partial \hat{p}^n_{nb}} \tag{4.15}$$

$$= -R^c_b \hat{R}^{nT}_b \tag{4.16}$$

$$= -\hat{R}^c_n, \tag{4.17}$$

$$\frac{\partial \hat{p}^c_{cm}}{\partial \hat{p}^n_{nm}} = \frac{\partial(-R^c_b p^b_{bc} + R^c_b \hat{R}^{nT}_b(\hat{p}^n_{nm} - \hat{p}^n_{nb}))}{\partial \hat{p}^n_{nm}} \tag{4.18}$$

$$= \frac{\partial R^c_b \hat{R}^{nT}_b(\hat{p}^n_{nm} - \hat{p}^n_{nb})}{\partial \hat{p}^n_{nm}} \tag{4.19}$$

$$= \frac{\partial R^c_b \hat{R}^{nT}_b(\hat{p}^n_{nm})}{\partial \hat{p}^n_{nm}} \tag{4.20}$$

$$= R^c_b \hat{R}^{nT}_b \tag{4.21}$$

$$= \hat{R}^c_n, \tag{4.22}$$

$$\frac{\partial \hat{p}^c_{cm}}{\partial \hat{\rho}^n_b} = \frac{\partial(-R^c_b p^b_{bc} + R^c_b \hat{R}^{nT}_b(\hat{p}^n_{nm} - \hat{p}^n_{nb}))}{\partial \hat{\rho}^n_b} \tag{4.23}$$

$$= \frac{\partial R^c_b \hat{R}^{nT}_b(\hat{p}^n_{nm} - \hat{p}^n_{nb})}{\partial \hat{\rho}^n_b} \tag{4.24}$$

$$= R^c_b \frac{\partial \hat{R}^{nT}_b(\hat{p}^n_{nm} - \hat{p}^n_{nb})}{\partial \hat{\rho}^n_b} \tag{4.25}$$

$$= R^c_b [\hat{R}^{nT}_b(\hat{p}^n_{nm} - \hat{p}^n_{nb})]_\times \ (\because (2.74)) \tag{4.26}$$

$$= R^c_b [\hat{R}^{nT}_b \hat{p}^n_{bm}]_\times \tag{4.27}$$

$$= R^c_b [\hat{p}^b_{bm}]_\times. \tag{4.28}$$

The same Jacobian matrix is found in [65]. Note that [65] uses the operator *tilde* for a skew operator $[\,]_\times$ and defines a camera coordinate in which the optical axis lies on the z-axis.

4.3 Camera calibration

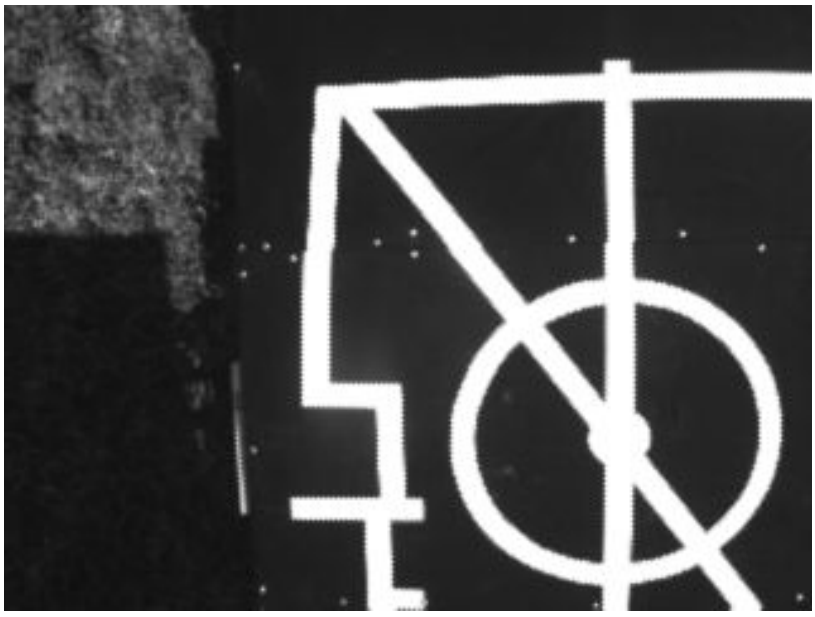
In this work, four camera intrinsic parameters and five distortion parameters are obtained through camera calibration. The four intrinsic parameters are often expressed as the camera intrinsic matrix as follows:

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.29}$$

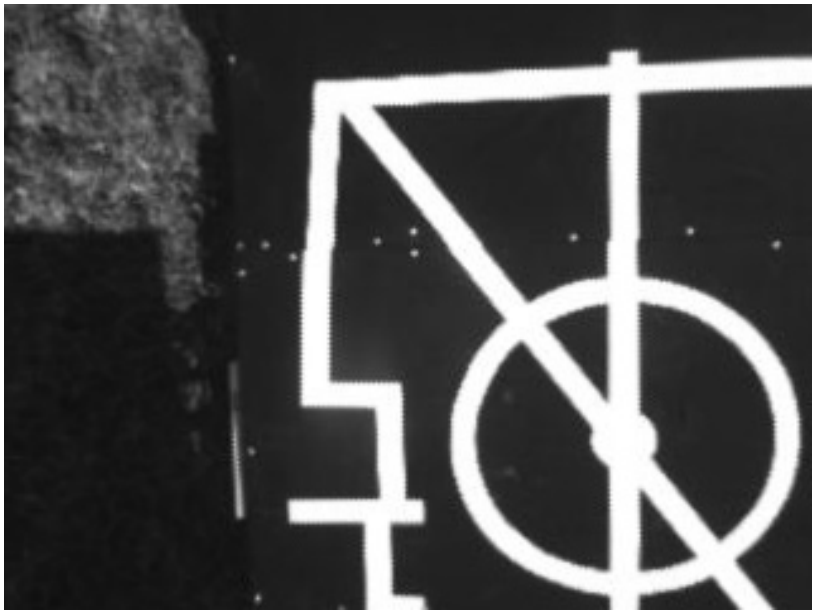
where f_x and f_y are the focal length of the camera, and c_x and c_y are the displacement of the center of coordinates on image plane. Let $\begin{bmatrix} u_p & v_p \end{bmatrix}$ denote the location on image plane if the pinhole camera were perfect (i.e. (4.4) without measurement noise). Then, they are expressed as follows using the distorted location $\begin{bmatrix} u_d & v_d \end{bmatrix}$:

$$\begin{bmatrix} u_p \\ v_p \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} u_d \\ v_d \end{bmatrix} + \begin{bmatrix} 2p_1 u_d v_d + p_2(r^2 + 2u_d^2) \\ p_1(r^2 + 2v_d^2) + 2p_2 u_d v_d \end{bmatrix}, \tag{4.30}$$

where $r = u_d^2 + v_d^2$, k_1, k_2, and k_3 are the radial parameters, and p_1 and p_2 are the tangential parameters. These two sets of the parameters model the radial distortion, which is caused as a result of the shape of lens, and the tangential distortion, which is caused by the misalignment of the lens and the image plane. The five parameters $(k_1, k_2, k_3, p_1, p_2)$ are the distortion parameters obtained through camera calibration. Figure 4.9 is an example of the image corrected using the distortion parameters described above.



(a) Raw



(b) Undistorted

Figure 4.9: Figures show the raw image obtained from the on-board camera 4.9a and the undistorted image 4.9b using the calibration parameters. The effects of the undistortion are most obviously seen at the top right corner of the image.

CHAPTER 5

RESULTS

5.1 Simulation results of portion detection

In this section, portion detection is evaluated with complete particle filter. The results of this section is reported in [101]. In simulation, vehicle position, velocity, attitude, and IMU bias states are estimated. For state estimation, GPS, sonar altimeter, magnetometer, and an IMU are used for measurements. The details of the estimator implementation are described in Chapter 3, and the flight simulator is described here [21]. In the simulation results presented here, the vision system is separated from the vehicle control where the inner and outer loop run at constant frequencies (100 Hz and 10 Hz). The on-board camera is simulated using OpenGL graphics using a resolution of 320 by 240 pixels. The vision system on an i7 desktop runs at approximately 20 Hz, which includes the measurement extractions and particle filtering using 3000 particles. The simulation enables waypoint tracking, which uses desired 3-D positions and velocities. This section evaluates the system with two scenarios: landing on a moving target, and chasing a moving target at low altitude. Both of these scenarios show the benefits of using particle filtering for target tracking of occluded imagery. The 2-D (X and Y) positions and velocities estimations are fed from the target tracker to the controller. The commanded descent speed for the landing is 0.5 ft/s and the desired altitude for following the target is set at 0.9144 meters (= 3ft). The simulated visual target is 1 meter square.

The results of the landing scenario are shown in Figure 5.1, 5.2, and 5.3. In this scenario, the initial vehicle altitude is 3.6576 m (= 12 ft), and the target is in the line of sight of the vehicle. At this altitude, the entire visual target is in view as shown in Figure 5.4-(a). The vision system finds the target soon after the simulation starts, and the vehicle

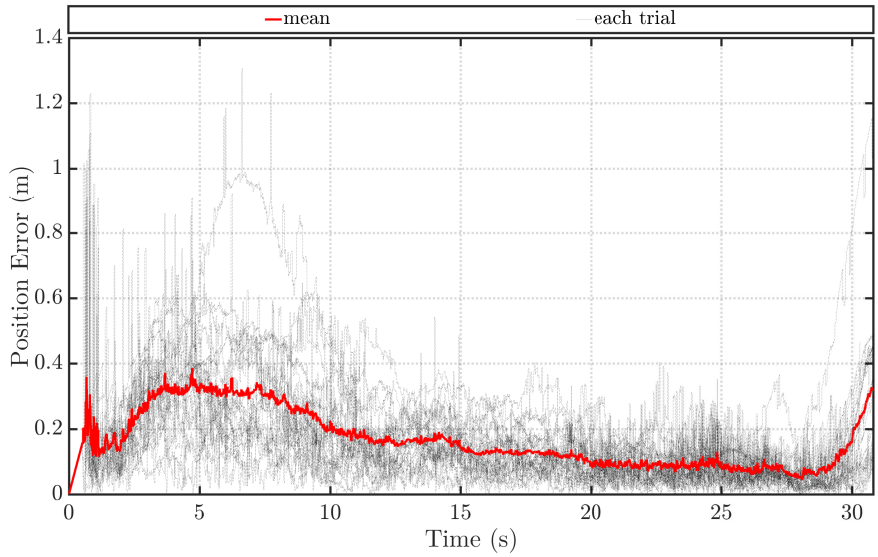


Figure 5.1: Figure shows the Euclidean distance errors in the position tracking of the vision system. The dotted black lines are the results of the 20 trials, and the red line represents their means.

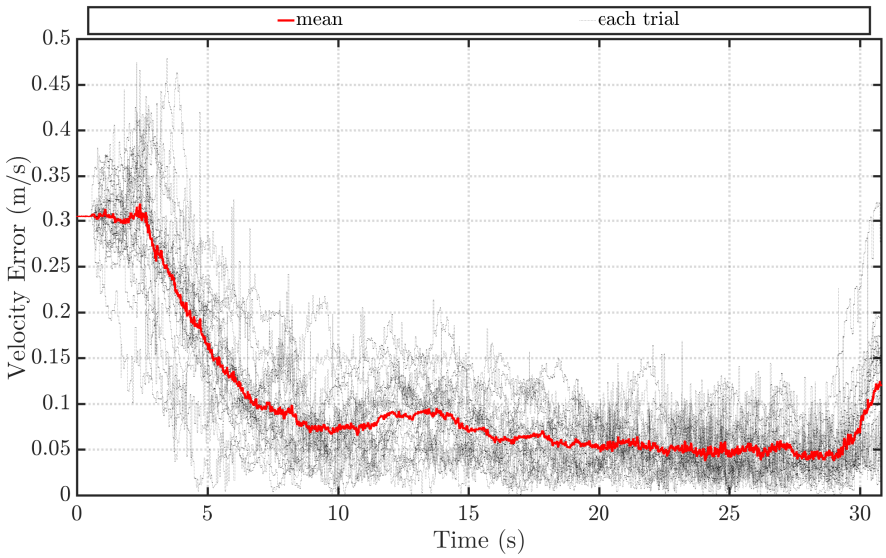


Figure 5.2: Shows the Euclidean distance errors in position tracking of the vision system. The dotted black lines are the results of the 20 simulated trials, and the red line represents the mean.

follows the target and lands. In this scenario, experiments evaluate how accurately the vision system can track the position and velocity of the moving target at different altitudes as the vehicle descends. The initial target speed is 0.3048 m/s (= 1 ft/s) with a small zero-

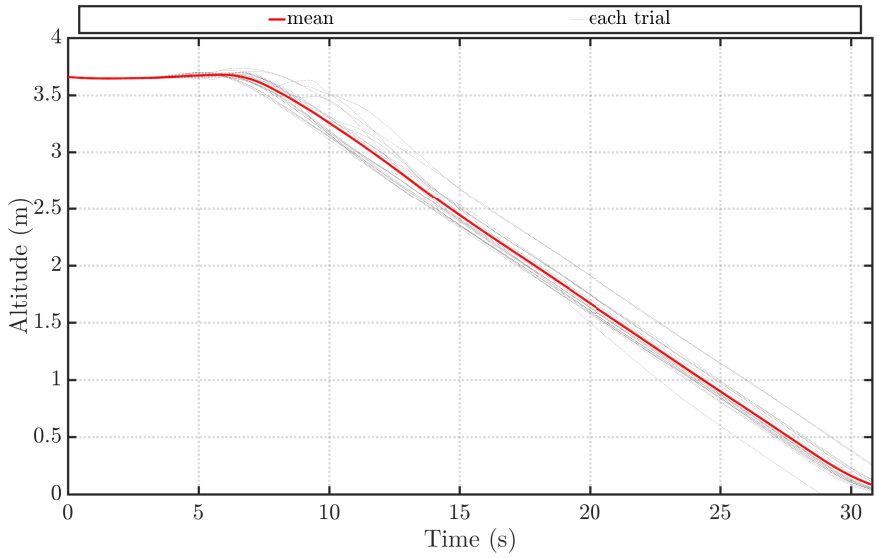


Figure 5.3: Shows the true altitudes of the aerial vehicle while landing. The dotted black lines are the results of the 20 simulated trials, and the red line represents their mean.

mean Gaussian noise ($N(0, 0.3048)$) added to the velocity of the target. The trials produce many interesting results. First, the vision system was able detect the target until the vehicle descended to around 0.2 to 0.4 m. Both of position and velocity errors monotonically increased below this altitude; thus, 0.2 m would be the lowest altitude the system can estimate the position and velocity of the target with good certainty. Second, the velocity tracking at high altitudes was not satisfactorily accurate. The velocity is not directly measured in the vision system; instead, it is estimated through sequences of images and the vehicle motion. As the vehicle descends and a more detailed visual target is available, the velocity estimation becomes more accurate. As the estimator becomes more straightforward Monte Carlo approaches, the performance of the velocity estimation is often outperformed by Kalman filter approaches. This trend remains the same for Rao-Blackwellized particle filter (RBPF) and extended Kalman particle filter (PF-EKF). Figure 5.4 shows the simulation scenes at the different altitudes and the corresponding on-board images. While hovering at low altitude, the vision system needs to estimate the location of the target most of the time from the partial information as shown in Figure 5.5. The chase scenario assesses the tracking

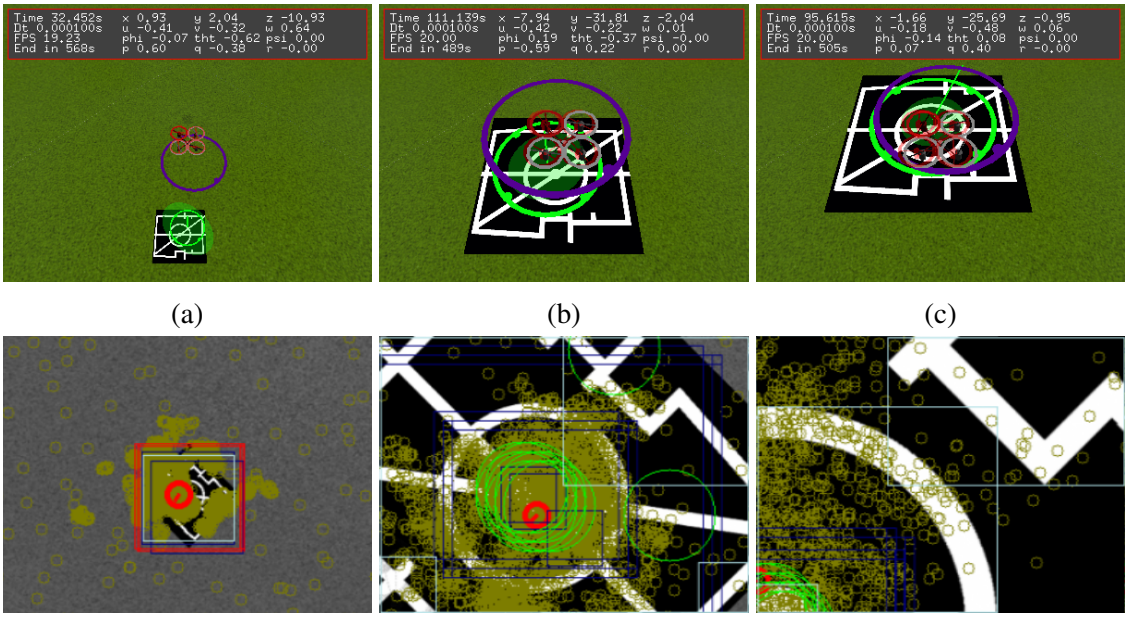


Figure 5.4: Figures on the top row show examples of the simulation scenes and the ones on the bottom row show the corresponding on-board images. The solutions of the vision system are overlaid on the on-board images. At high altitudes such as (a) and (d), the entire visual target is seen and the system detects the target with method 1 described in the previous section. The red rectangle indicates the measurement method 1. The blue rectangles on (e) and (f) are the measurements with method 2, and the green circle in (e) and (f) are the measurements of method 3.

accuracy at an altitude of less than 1 meter. At low altitudes, the entire target does not fit into the on-board image even when the UAV flies over the center of the target. During this scenario, to estimate the location of the target only partial information is available, as seen in Figure 5.5. In this scenario, the UAV starts at 0.9144 m (= 3 ft) over the center of the target. The target moves at 0.6096 m/s (2 ft/s) with a constant turn rate. A small process noise is added to the turn rate and the speed of the target. In Figure 5.6, the true and estimated vehicle and target positions are displayed. Figure 5.7 shows the tracking errors and the threshold equivalent to the 2σ expressions, which means that when the 2σ bounds are inside the threshold, the solution of the vision system is used for the closed-loop system. The simulation results show that the vehicle was able to track the target for more than 100 seconds without going below the threshold.

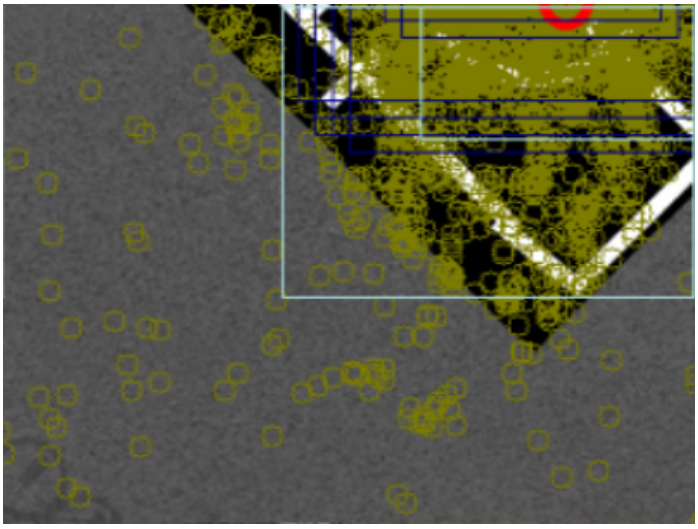


Figure 5.5: Figure shows the outputs of the vision system overlaid on a simulated airborne image. When flying at a low altitude, there are frequent occasions that the vision system needs to estimate the target location from a partial target. The red circle is the estimated target center. The light blue rectangles are interesting regions extracted by the contour trees, and the dark blue rectangles are the matched locations using these regions. If multiple locations have the NCC beyond the threshold, all of them are used as measurements; thus, one interesting region may generate multiple measurements.

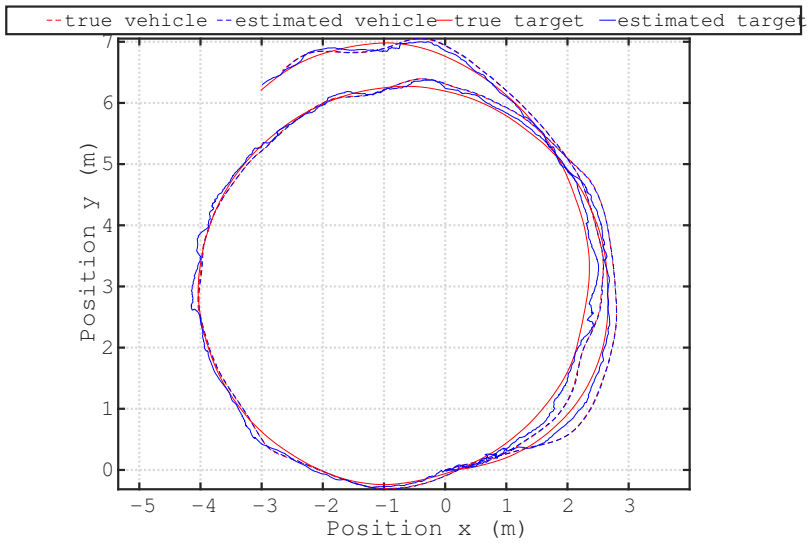
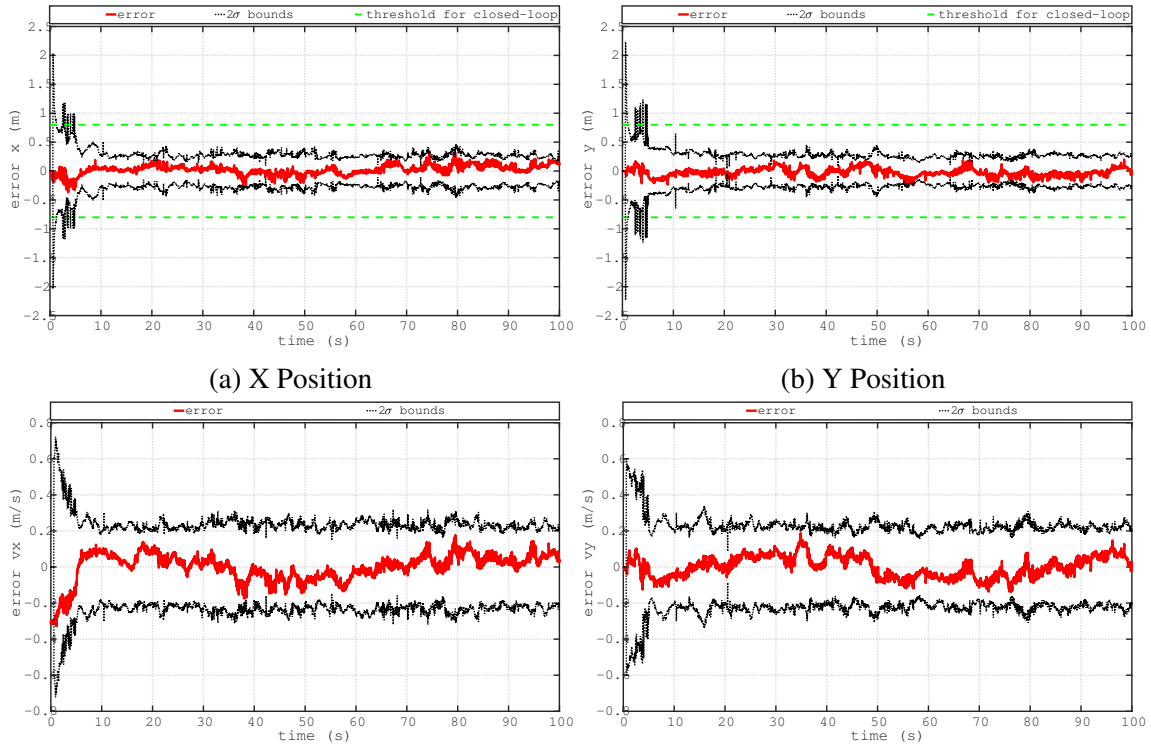


Figure 5.6: Figure shows the true and estimated position of the vehicle and target while the vehicle chases the target at 0.9144 m (= 3 ft). The dimensions of the target are 1.0668 by 1.0668 m (= 3.5 ft) and the chase altitude was chosen to have the entire target occluded during the maneuver.



(a) X Position

(b) Y Position

(c) X Velocity

(d) Y Velocity

Figure 5.7: Figures show the estimation errors in the position, velocity, and 2σ bounds. The covariance of the position estimation are used as the criteria for closed-loop tracking as described in Chapter 3.3. When the 2σ bounds are the inside of the threshold lines, the solution of the vision system is used for the closed-loop tracking.

5.2 Flight test results

5.2.1 Flight Tests with DJI Matrice

Flight tests were performed to test the proposed target tracking algorithm using a DJI Matrice-100[1] (Figure 5.8-(a)). The Matrice-100 is a fully programmable UAV that can be customized using the DJI SDK[2]. DJI provides Robotic Operating System (ROS) nodes to gather sensor information, send commands, and receive images from the X3 gimbal camera. The vehicle houses the N1 flight controller which provides IMU, barometric, magnetometer, and GPS data, and a NVIDIA-based Tegra-K1 computer (Manifold) for extra computational processing. An additional sensor was added to provide an above ground

[1] `http://www.dji.com/matrice100`

[2] `https://github.com/dji-sdk/on-board-SDK-ROS`

(a) DJI Matrice (b) DJI Matrice flying over a marker

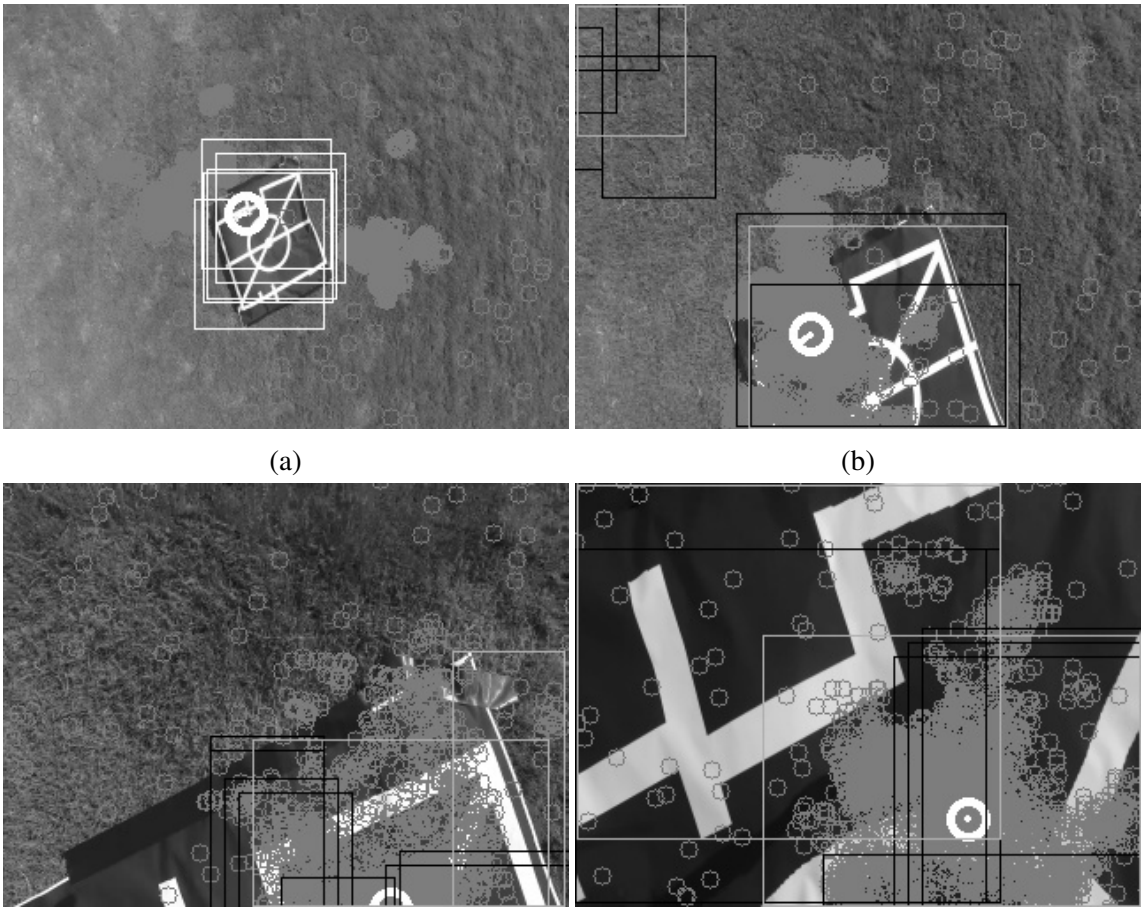
Figure 5.8: Figure (a) shows our flight test vehicle DJI Matrice 100, and (b) shows the vehicle is hovering over the target for flight testing.

level (AGL) measurement. The outdoor version of the TeraRanger One[3] is used which provides altitude measurements at rates up to 1000Hz. The proposed system is implemented in the Georgia Tech UAV Simulation Tool (GUST). GUST is a software framework that is used for simulating UAVs and also houses the flight code used on different UAVs from helicopters to small quadrotors [121]. It also provides a Ground Control Station (GCS) interface for flight test operations. A ROS node was created to interface GUST with the DJI vehicle. GUST and ROS are both run on-board the DJI Manifold computer. The GUST-Link node is the interface between ROS and GUST. It subscribes to the DJI provided node and sends sensor and target tracking outputs via UDP packets to GUST, which runs the trajectory generation, EKF navigation, and controller. Navigation states and flight control commands are sent back over UDP packets to the GUST-link node, which publishes the navigational data for the target tracking node and control commands for the vehicle to follow. GUST runs a model-inversion controller with an inner/outer loop structure [122]. Through the DJI-SDK, individual motor commands are not accessible by the user. The lowest level of control allowed is attitude commands, while using higher level position-based commands is recommended. The control commands are directly pulled from different lev-

[3] `http://www.teraranger.com/products/teraranger-one/`

els in control loops. For this work the position-based commands were used.

To test UAV operations using both GUST and ROS, a different simulation environment was needed. The ROS package, the hector quadrotor [123], was modified sensor and control architecture-wise to simulate the DJI Matrice-100. TGUST and ROS are both run on-board the DJI Manifold computer. Sensor information and outputs from the target tracker are passed from ROS to GUST via UDP packets where GNC algorithms are run and the navigational states and control commands are passed back. The Target tracker node subscribes to both the GUST navigational states, X3 camera stream, and outputs position, and velocity of the target. The hector quadrotor model uses Gazebo as a graphical and physics engine. In the flight test, the UAV landed on a static target texture whose dimension was 1.0668 m by 1.0668 m. The X3 camera provided 640 pixels x 480 pixels images to the vision system. In the flight test, 3000 particles were used, and the on-board images were scaled to 320 pixels by 240 pixels. The target tracker ran at approximately 20 Hz. Since the heading of the target is not estimated through the motion of the target in this case, the heading of the target was measured and directly given to the vision system. The UAV first hovered over the target using a GPS signal. The target tracker started and converged to the target location, then the target waypoint was switched from a GPS to the output of the target tracker when the vehicle was at approximately 3.5 m above the ground. Figure 5.9 shows the on-board images obtained in the flight test while the vehicle was attempting landing. The measurements and particle locations are overlaid on those images. As shown in Figure 5.9-(a), the detection rate at high altitude is reliable. However, the position of the marker extracted with pure template matching is not very accurate. This is the first test with real environment images (not simulation), and the results of this experiment is a deciding factor for the introduction of the machine-learning-based method at high altitude. The system saw more false positives than the simulation, as shown at the top left corner of Figure 5.9-(b). The region found at the top left corner was used for the NCC template matching, and generated a false positive on the top left. The candidate detection at corners



(a) (b)

(c) (d)

Figure 5.9: Figures show the on-board images obtained in the flight test during a landing maneuver. The results of the vision system are overlaid on the on-board images. From a high altitude to right before the touch down, the target position was available. The same notations explained in Figure 5.4 are used. Grayscale images are used to save a computational power.

and edges of images ignore the size matching because what vision observes may be a part of the marker, not an entire one. However, a certain number of pixels needs to be observed, and the number needed to be increased for flight tests. In the flight test, some contours were found nowhere near the true target locations, but they typically displayed a small NCC, and not many particles used these values for an update. One false positive cannot drive all the particles because only the particle that consistently has a high likelihood survives. Even when only part of the target is available as shown in Figure 5.9-(c) and is even more occluded due to the low altitude seen in Figure 5.9-(d), the system was able to track the target

while landing.

5.2.2 Flight tests with RMAX helicopter

A modified Yamaha RMAX helicopter UAV (shown in Figure 5.10) is used for flight tests. The take off gross weight is approximately 200 pounds, and it is equipped with a flight computer, an IMU, a differential GPS receiver, a laser range sensor, a 3-axis magnetometer, and a camera. The on-board computer records the states of the vehicle as well as on-board camera images. The baseline flight controller is an adaptive neural network controller following a trajectory. The neural network has 18 inputs and 7 outputs corresponding to the six-rigid-body degrees of freedom and the rotor RPM [122]. Georgia Tech UAV Simulation Tool (GUST) [124] is the flight control software utilized for this flight tests, which is written in C/C++. The Marine Advanced Research 16 WAM-V USV is used as the vehicle carrying the landing platform. The vehicle is capable of cruising at its maximum speed of around 10 knots for about 3 hours and has a specified payload of 250 lbs. A modular 8x12-ft landing platform is built and attached to the top of the vehicle during testing, essentially all of which is usable for landing the 200 lb RMAX helicopter. This is despite the helicopter and platform payload being well over the manufacturer's conservative maximum payload specification. The vehicle features an articulated suspension system which is beneficial for stabilizing the top platform in sea state. However, as the platform and helicopter weight are over the specified payload limit of the WAM-V, the damper pistons were stiffened to prevent over-flexure of the suspension system during landing. This is done to keep the platform from over-flexing and potentially allowing the helicopter to roll over on or after landing. The particle filter approaches are tested with the recorded data, and Figure 5.11 and 5.12 show the on-board camera images with the annotations displaying the results of the detection and the RBPF. The same annotations as in Figure 5.4 for detection of targets are used, except that the output of the filter is drawn as a green circle with estimated size of the target. The suggested detection algorithm is validated with the actual on-board camera

images, and the RBPF and PF-EKF successfully estimated the position of the marker even when only a partial image is in view. RBPF more stably estimates the position of the marker. These results match with the simulation results discussed in Section 5.5 for static targets. Two experimental data are recorded. In one experiment, the vehicle lands on a deck, and in the other experiment, the vehicle passes through the deck at low altitude. Figure 5.11 shows the results of the landing case, and Figure 5.12 shows the results of the low-pass approach case. The lowest altitude that observed the marker is 2.5 ft above target. Considering that the marker is 8x8 ft (square part of landing platform, which is 8x12-ft) and the field of view of the camera is approximately 45 degrees, the entire marker is observed until 9.7 ft above the target. The portion detection extends the detectable range by 7.2 ft. In simulation, SIFT-based feature matching is turned off when entire marker can not be observed. However, in flight tests, there are more features to be used for matching, and the SIFT-based method is better at finding at portion of the marker in flight tests than in simulation. Therefore, the SIFT-based method is only turned off at 4 ft above the target, at which the marker is inevitably occluded.

Figure 5.10: Figure shows the RMAX helicopter landing on a ship on the ground.

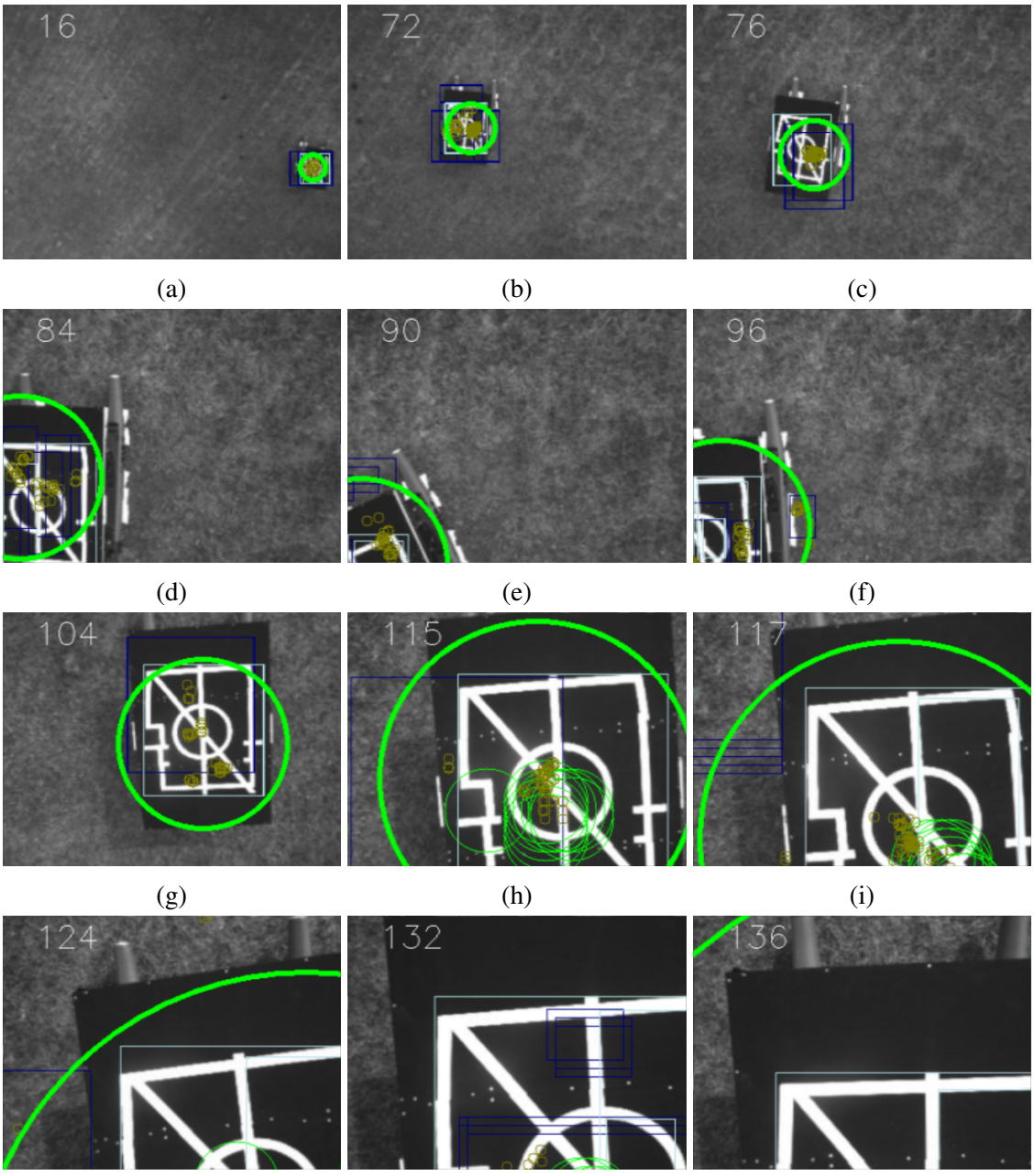


Figure 5.11: Figures show the images obtained in the flight test while the RMAX helicopter lands on a navy deck pattern. The results of the detection algorithms and the RBPF are overlaid.

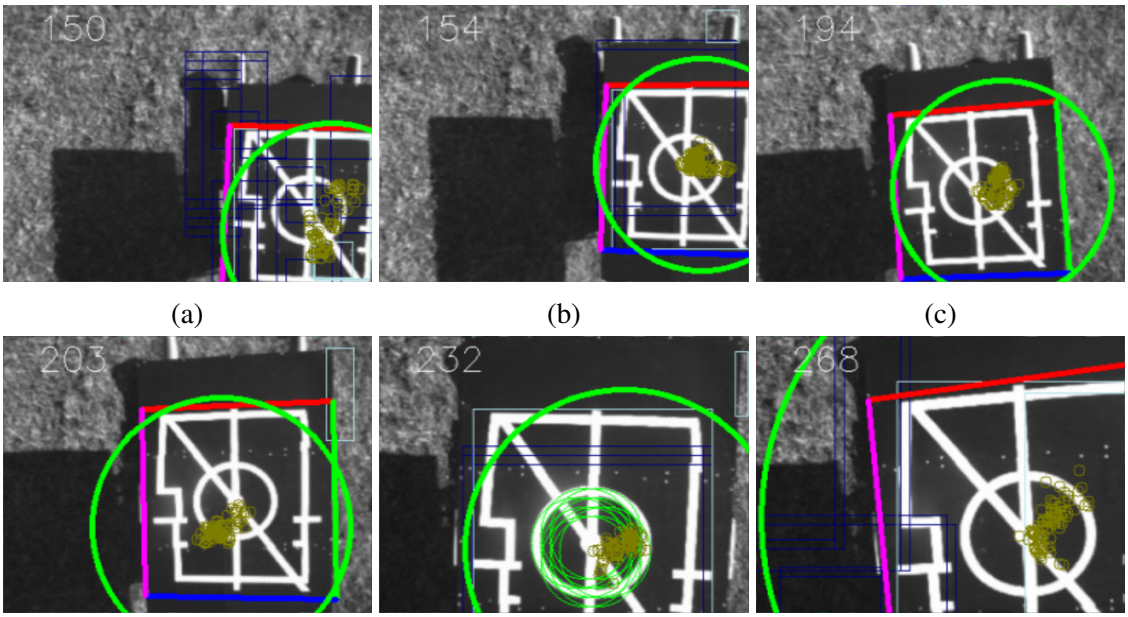


Figure 5.12: Figures show the images obtained in the flight test while the RMAX helicopter hovers low over a navy deck pattern. The results of the detection algorithms and the RBPF are overlaid.

5.3 Validation of marker-aided EKF

5.3.1 Chi square test

The statistics of the χ^2 is used as a measure to test accuracy of fit of observed data to theoretical distributions. For a n-degrees of Gaussian-distributed random variable $x \sim N(\mu, \Sigma)$, the χ^2 is defined as follows:

$$\chi_n^2 = (x-\mu)^T \Sigma^{-1} (x-\mu). \tag{5.1}$$

The distributions of the χ^2 is independent from the property of x, and the comparison between the ideal χ^2 distribution and the observed distribution provides a correctness of the model, implementation, and more. Figure 5.13 shows an example of the CDF of χ^2 distributions. The details of the χ^2 test are found in Appendix B of [61]. One method to verify

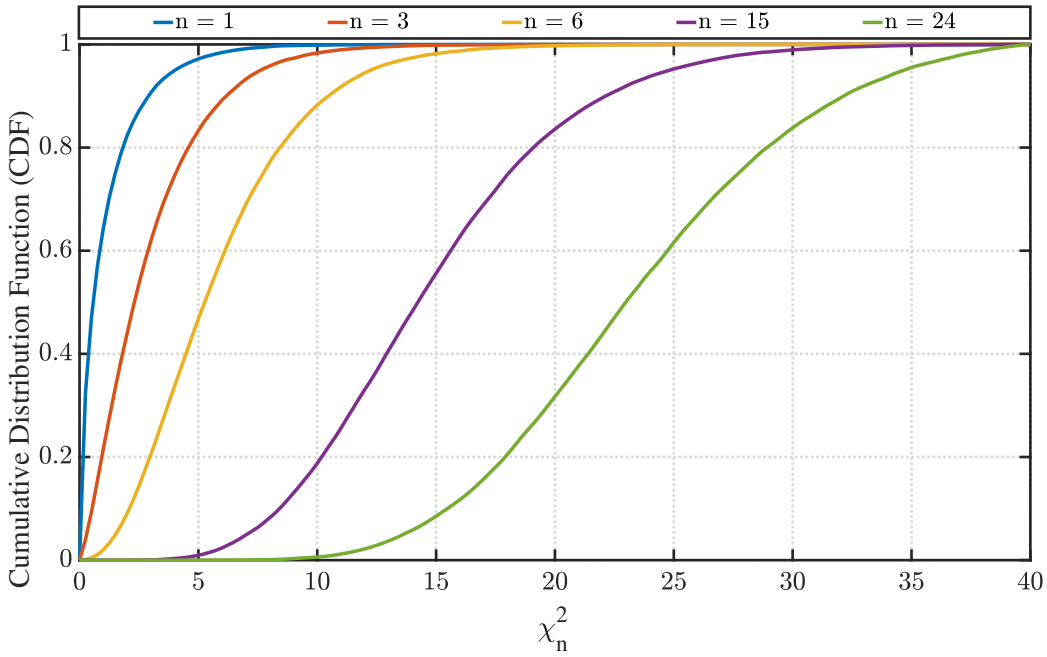


Figure 5.13: CDF of chi-square distributions with various degrees of freedom.

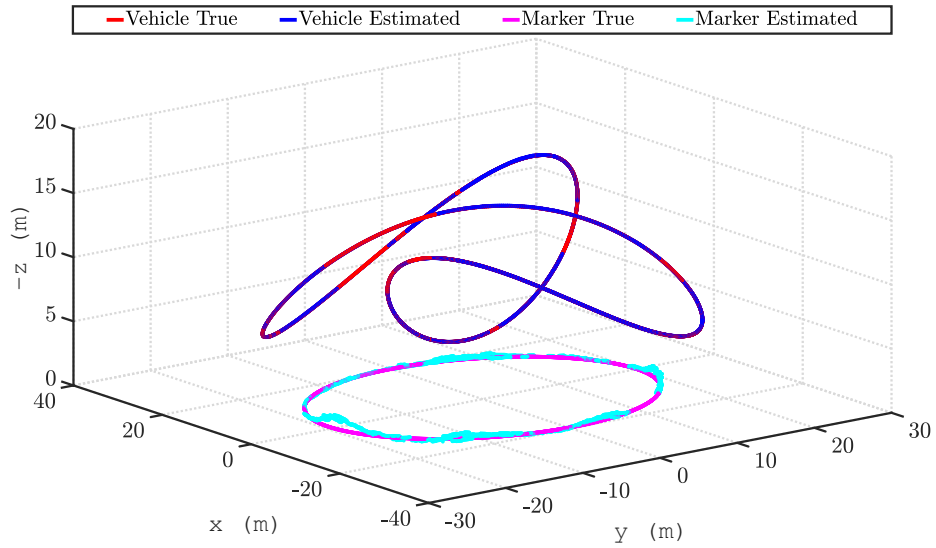
the implementation of a Kalman filter is to use knowledge of the statistics of the innovation, or measurement residuals. The innovation is the difference between a measurement and an

estimated output, which, in a linear system, is defined as follows:

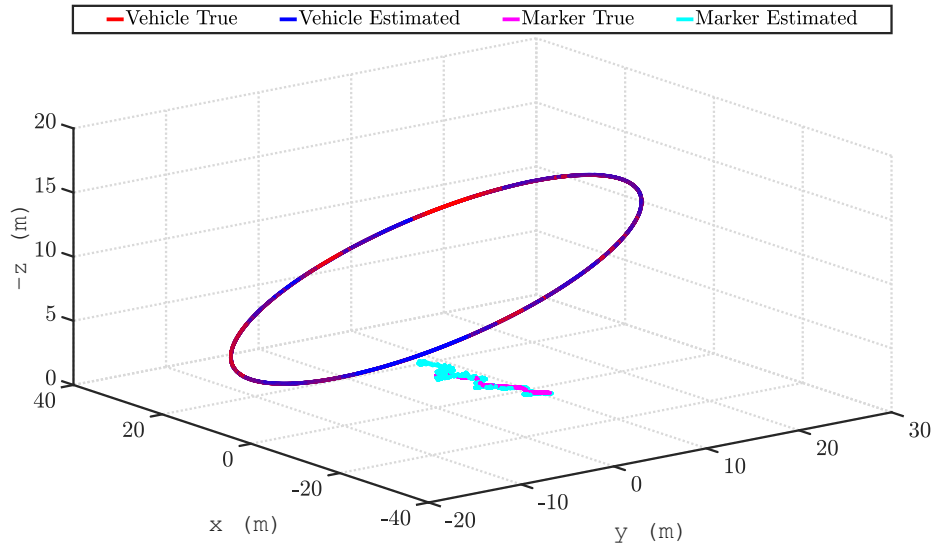
$$\epsilon_k = z_k - y_k = z_k - H_k \hat{x}_k^- . \tag{5.2}$$

The innovation is a zero-mean, white, stochastic process with a known covariance. The details of the stochastic properties of the innovation are described in Chapter 10.1 of [79]. The covariance of measurement residuals can be obtained through estimated error covariance of an EKF as shown in (3.101). In order to validate the implementation, synthetic trajectories of the vehicle and marker are generated. Figure 5.14 shows examples of the synthetic trajectories used for validation. The sinusoidal functions (see Appendix C) generate trajectories of the vehicle, which are either circular or Lissajous trajectories. The dynamics of the marker are expressed as either a circular model or random walk model. The pseudo visual measurements are generated by using the positions and attitude of the vehicle and marker; thus, in this validation, the visual measurements are perfectly zero-mean and Gaussian-distributed. The update rate of the visual measurements is set to 20 Hz. Figure 5.15 show an example of the residual of the visual measurements. Although the behavior of the measurements and their estimated error covariance are almost Ergodic, and the statistics of a single run are nearly identical to the statistics across various runs, Monte Carlo simulations are conducted to further verify the implementation. Figure 5.16 shows the results of the Monte Carlo simulation with 200-runs of the 40-second simulations. This figure shows the expected cumulative distribution function (CDF) of 3-degree (red), which is computed from ideal normally-distributed random variable. The statistics of the measurement residuals have a solid match with the expected curve.

The estimated errors are evaluated using the same Monte Carlo simulation. In order to obtain a good statistical match in this Monte Carlo simulation, it is important to make sure all the states are observable. As is reported in multiple studies (e.g. 12.7 of [71]), in EKF estimating the biases of the gyroscope and the attitude of the vehicle, at least one-axis of



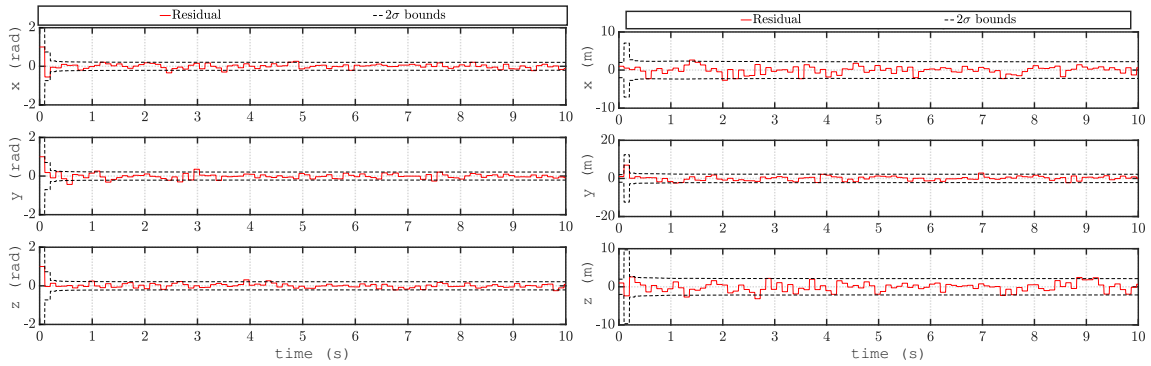
(a) Lissajous (vehicle) and circle (target)



(b) Circle (vehicle) and random walk (target)

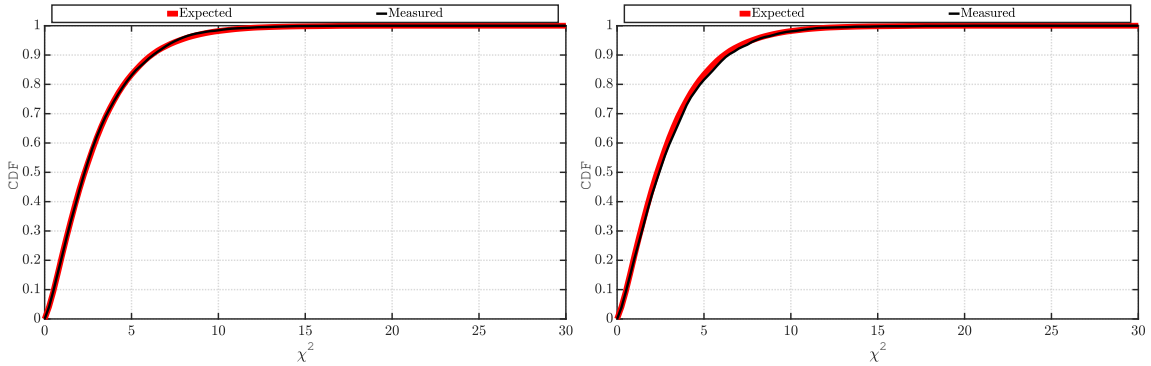
Figure 5.14: Examples of trajectories used for Monte Carlo simulations

the motion needs to be excited to make a fully observable state space. In this experiment, initialization parameters, which include the coefficients of sinusoidal functions, are randomly chosen. Figure 5.17 shows the results of the Monte Carlo simulation. 200-runs are used for this Monte Carlo simulation. In this experiment, the mean of the diagonal entries of the estimated error covariance matrix over the 200 runs is compared with the covariance of the estimation error across 200 runs. Figure 5.17 shows that all the marker states show a great match between the estimated covariance and the measured covariance. Theoretically,



(a) Angle (b) Position

Figure 5.15: Residuals and their 2 sigma bounds of marker measurements



(a) Angle (b) Position

Figure 5.16: Chi square test of the measurements of marker angles and positions

the estimation error should be zero-mean; however, some results clearly show the correlation of the mean estimated error and time. Through this experiment, the author realized that the choosing various radii of the functions, denoted A in the trajectory model (C.3), is more effective than varying the frequencies of the function, denoted B in (C.3). Therefore, the variation of the frequencies and other parameters are not great enough to make a perfect zero-mean error distribution. At some specific time, many trials tend to have a large error (e.g. arguments of sin are $\pm\pi/2$). Figure 5.18 shows the chi square tests of this experiment. This would be good enough to validate the implementation of the filter, but some states are slightly pessimistic. Particularly, the bias estimations tend to be statistically incorrect because of the properties of AR(1).

Another interesting thing to note is the correlation of the estimated errors. Although the chi square test (Figure 5.18) validates the statistics of the estimations errors and their

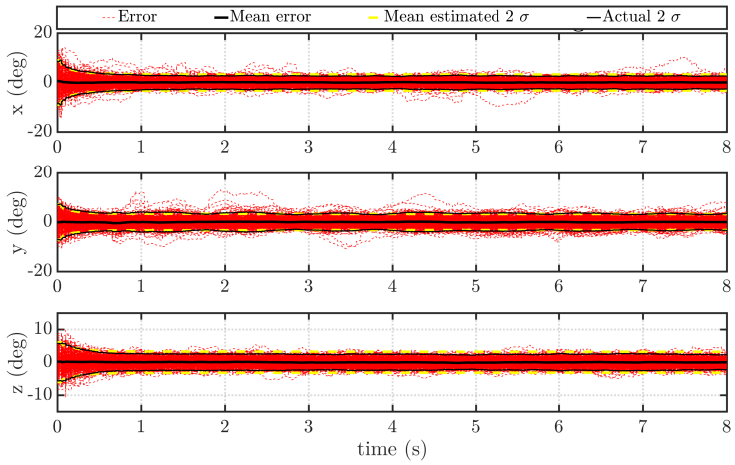
covariance, but they are not plotted. Figure 5.19 shows the time history of the off-diagonal entries of the estimated error covariance. The time history is computed by taking the mean of the 50 runs of the simulations. This only shows the entries in the upper triangular matrix, and the labels are explained in (5.3).

$$P = \begin{bmatrix} [1 \leftrightarrow 1] & [1 \leftrightarrow 2]_1 & [1 \leftrightarrow 3]_2 & [1 \leftrightarrow 4]_3 & & \cdots & [1 \leftrightarrow n]_{n-1} \\ & [2 \leftrightarrow 2] & [2 \leftrightarrow 3]_n & [2 \leftrightarrow 4]_{n+1} & \cdots & & [2 \leftrightarrow n]_{2n-3} \\ & & [3 \leftrightarrow 3] & [3 \leftrightarrow 4]_{2n-2} & \cdots & & [3 \leftrightarrow n]_{3n-6} \\ & & & & \ddots & & \\ & & & & & \ddots & \\ & & & & & [n-1 \leftrightarrow n-1] & [n-1 \leftrightarrow n]_{(n^2-n)/2} \\ & & & & & & [n \leftrightarrow n] \end{bmatrix} \tag{5.3}$$

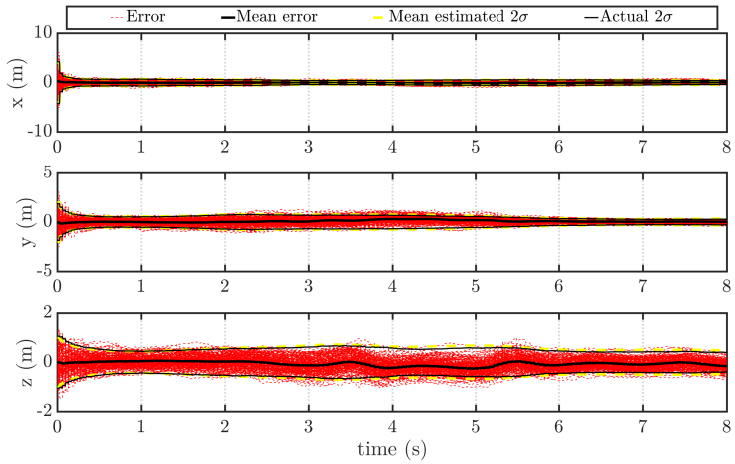
For example, $[20 \leftrightarrow 23]$ is the covariance of 20-th state and 23-rd state in (3.69), which is the y component of the marker position and the y component of the marker velocity. The subscripts are only added to the upper triangular entries. This is the x-axis of Figure 5.20. The covariance between the position of the marker and velocity tends to be large. Firstly, the measurement error covariance of the vision is greater than any other sensor. Since the angle measurements are expressed in radius, the states most significantly affected by the position measurement show large values of covariance. Secondly, there is no deciding sensor for marker velocity. For example, the position and the velocity of the vehicle may have a larger correlation. However, in this work, the velocity of the vehicle is directly measured through GPS, which is assumed to be independent from position measurements. On the other hand, the velocity of the marker is estimated only by sequentially observing the marker position. In fact, when correlation is compared across all the states, the correlation between the marker position and velocities are greater than any other states. Figure 5.20 displays the mean correlation of all combination of the states. The correlation of the errors corresponding to the state i and j are computed as follows:

$$\text{corr}(x_i, x_j) = \frac{\text{cov}(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}} = \frac{P(i,j)}{\sigma_{x_i}\sigma_{x_j}}. \tag{5.4}$$

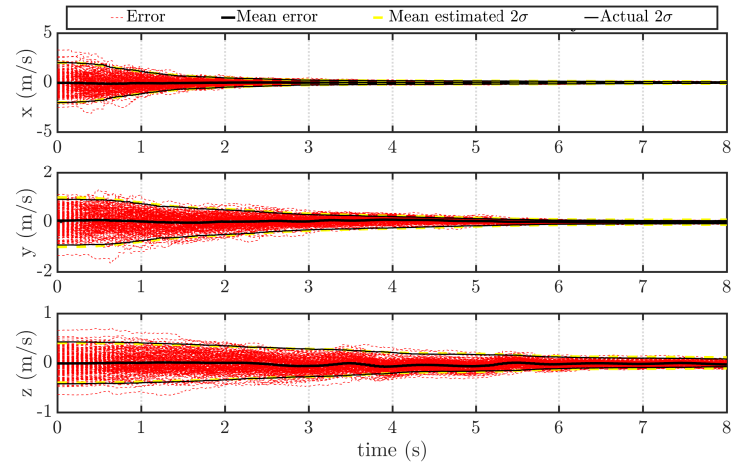
The correlation between the position of the marker and the corresponding axis of the velocity (19-22, 20-23, and 21-24) are all greater than 0.6. Although the position and the velocity of the vehicle (4,5,6 and 7,8,9) are also larger than average, they are smaller than those of markers because of the addressed reason.



(a) Euler Angle



(b) Position



(c) Velocity

Figure 5.17: Monte Carlo simulation results across 200 runs. The red lines are the errors of each run. The yellow lines are computed from the estimated error covariance matrix, and the skinny black lines are the variance of the errors across 200 runs. The wide black lines are the mean of the errors across 200 runs.

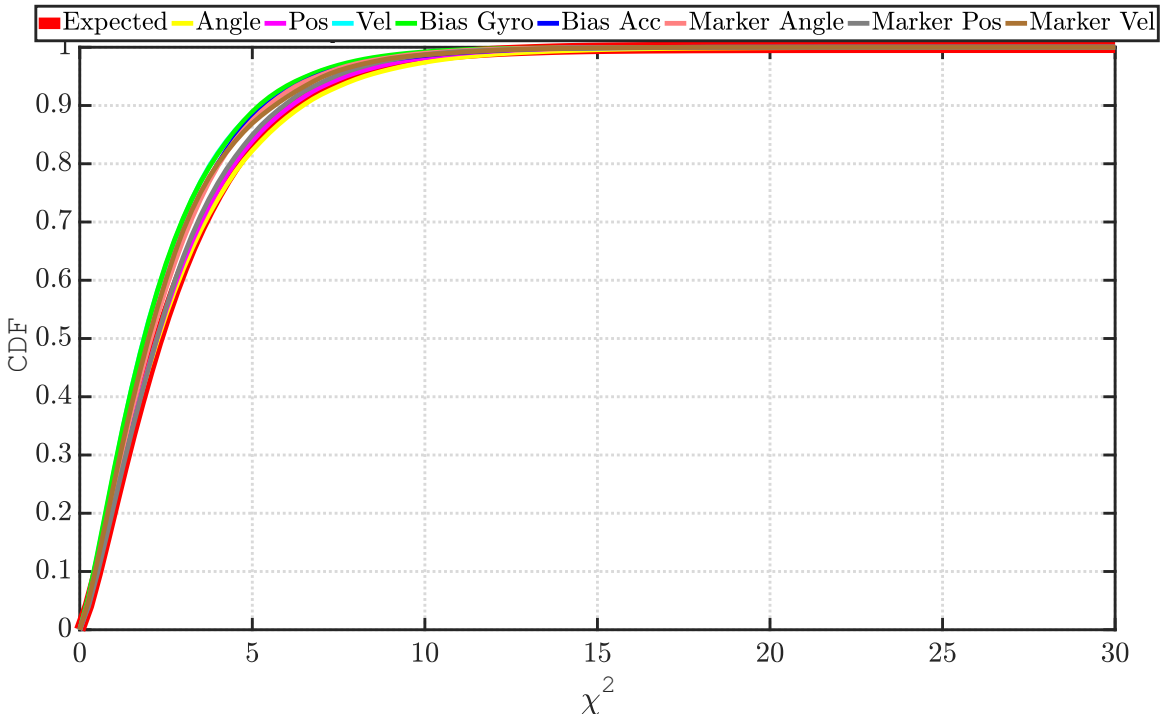


Figure 5.18: Figure shows the Chi square test of Monte Carlo simulation using 200 runs.

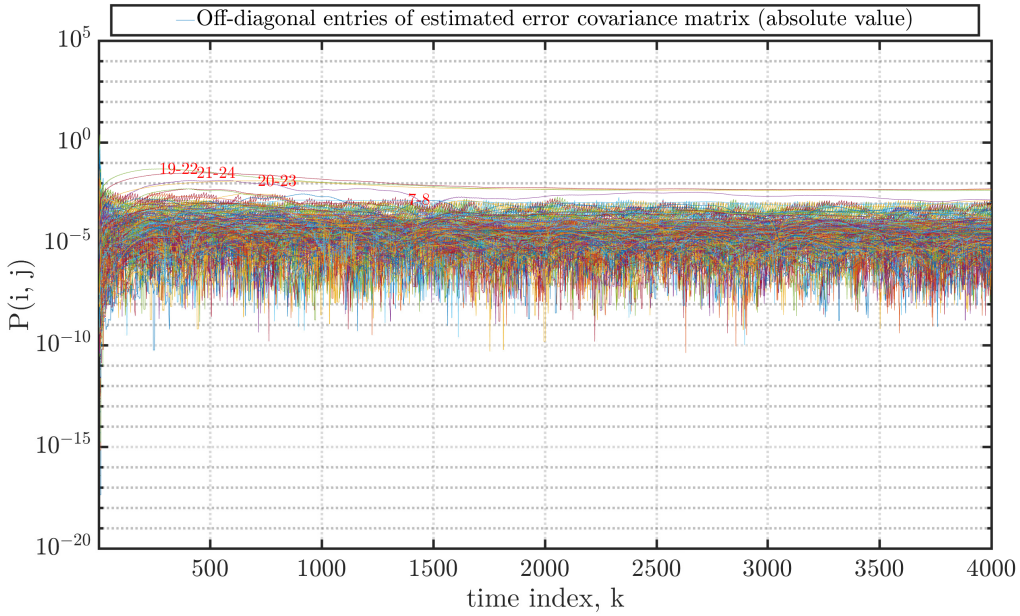


Figure 5.19: Time history of the estimated error covariance matrix. Figure shows the mean of the off-diagonal entries across 50 runs. The index is explained in (5.3). The covariances of marker position and velocity have the large values.

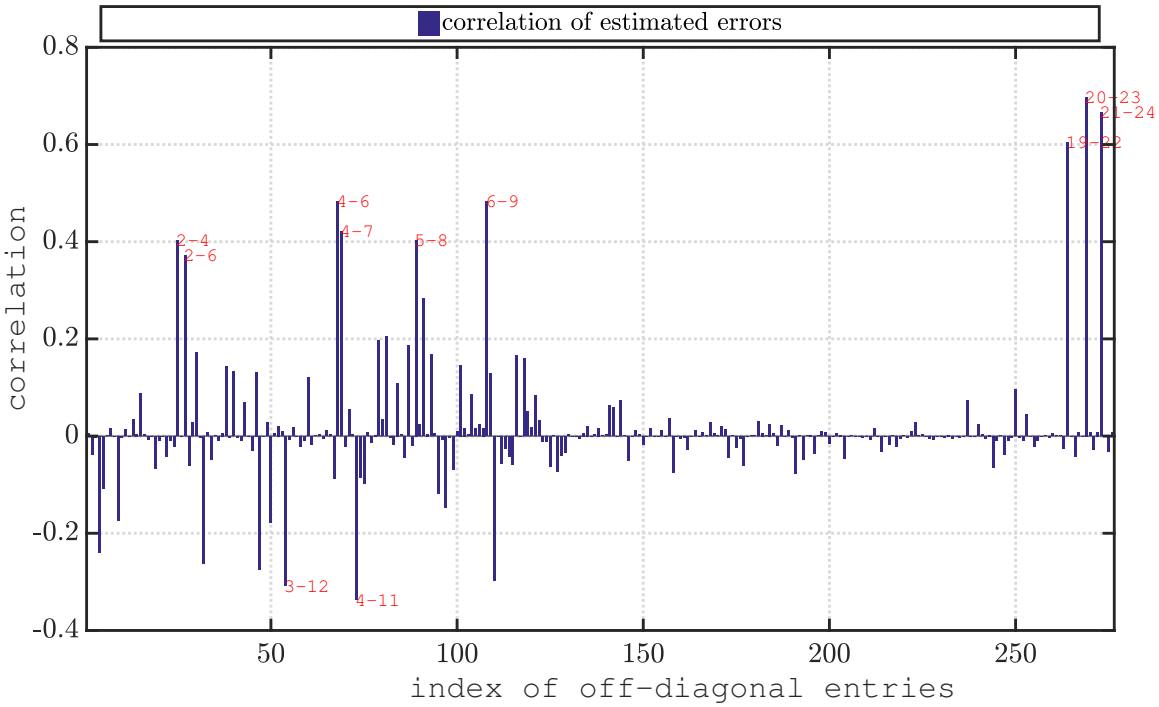


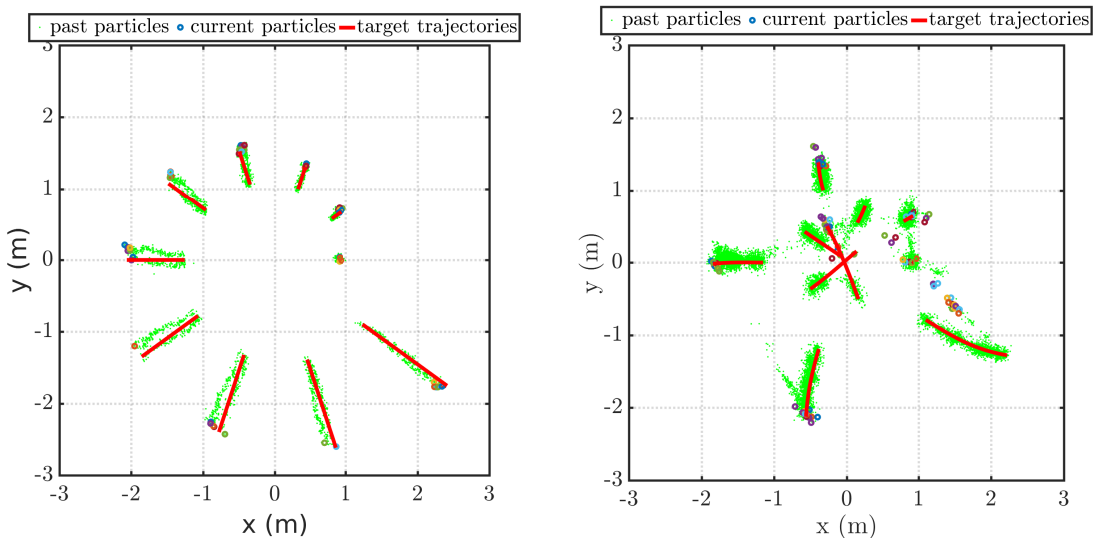
Figure 5.20: Figure shows the correlations of the estimated errors. The index is explained in (5.3). The correlation between the marker position and velocity are greater than the correlation of other states.

5.4 Tracking multiple targets

The extended Kalman particle filter (PF-EKF) and Rao-Blackwellized particle filter (RBPF) estimate the distributions of the estimated states. When tracking a known number of targets, and the specific target generating a measurement is known, computing the states of each target from the distribution is straightforward; however, this is not often the case for UAV operation. In most cases, the number of targets is not known, and whether a measurement is generated by specific one of the targets or a false positive is also not known. Specifically, when the navigation system is used for vision-based landing, limiting the possible landing locations to a single site may not be recommended; instead, estimating the states of all the possible targets in view is desired. That is why, in this section, the capability to track a unknown number of targets is evaluated. One of the most common methods for tracking an unknown number of targets is to vary state space dimensions based on the known probability of detection, process models, or environmental factors [125], [126], [127]. This presented work utilizes the approximated version of the variable-state-space approach for a regular extended Kalman filter (EKF). The EKF introduced in 3.2 tracks only one marker. In order to compare the performance with particle-filter-based estimators, the EKF in this section is modified to be able to track multiple markers by dynamically varying the state space dimension. The EKF utilized in this section holds the following state vector:

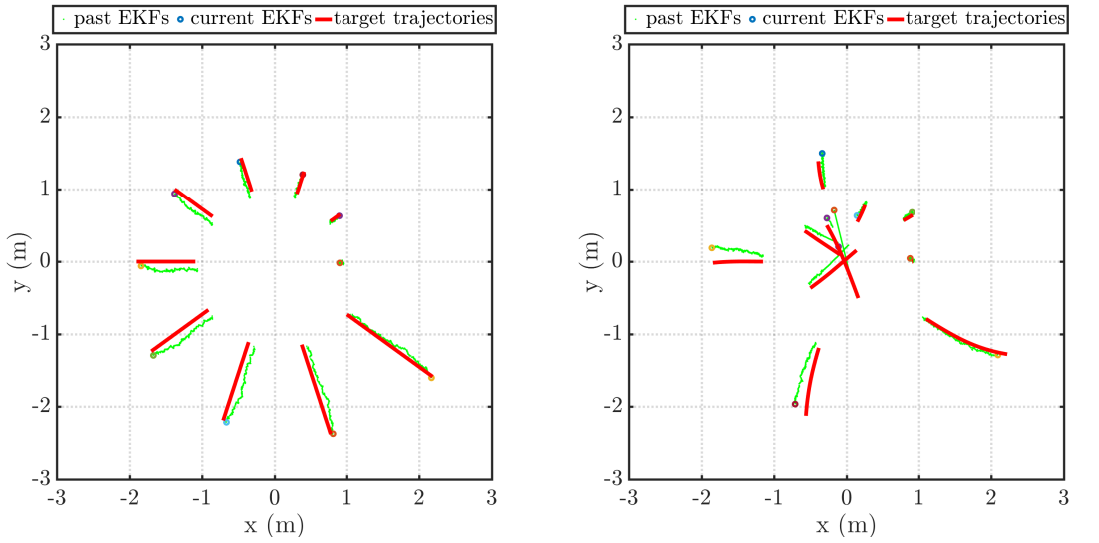
$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\rho}_b^n & \hat{p}_{nb}^n & \hat{v}_{nb}^n & \hat{b}_{gyro} & \hat{b}_{acc} & \begin{bmatrix} \hat{\rho}_m^n & \hat{p}_{nm}^n & \hat{v}_{nm}^n \end{bmatrix}_1 \cdots \begin{bmatrix} \hat{\rho}_m^n & \hat{p}_{nm}^n & \hat{v}_{nm}^n \end{bmatrix}_m \end{bmatrix} \in \mathbb{R}^{15+9m}, \tag{5.5}$$

where m is the number of targets. The Mahalanobis distance is used to separate targets. When a detection is found outside of the Mahalanobis distance threshold, a new target (nine more states) is added to the state vector. When an estimated error covariance of the target exceeds a certain value, the corresponding target is eliminated from the state vector. However, this approach is implausible for the particle-filter approaches of this work. The most expensive operation of EKF is the propagation, which is $O(n^3)$, where n is the dimension of the state vector. The computational cost of the other operations is also increased as the



(a) Radial target trajectories (b) Random target trajectories

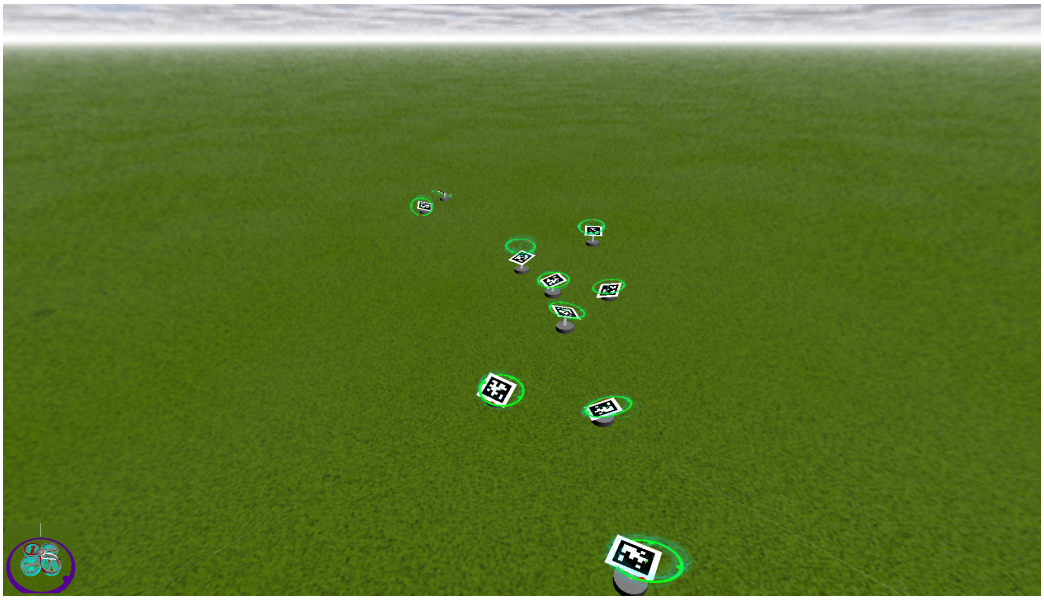
Figure 5.21: Figures show the trajectories of the targets and particle distributions.



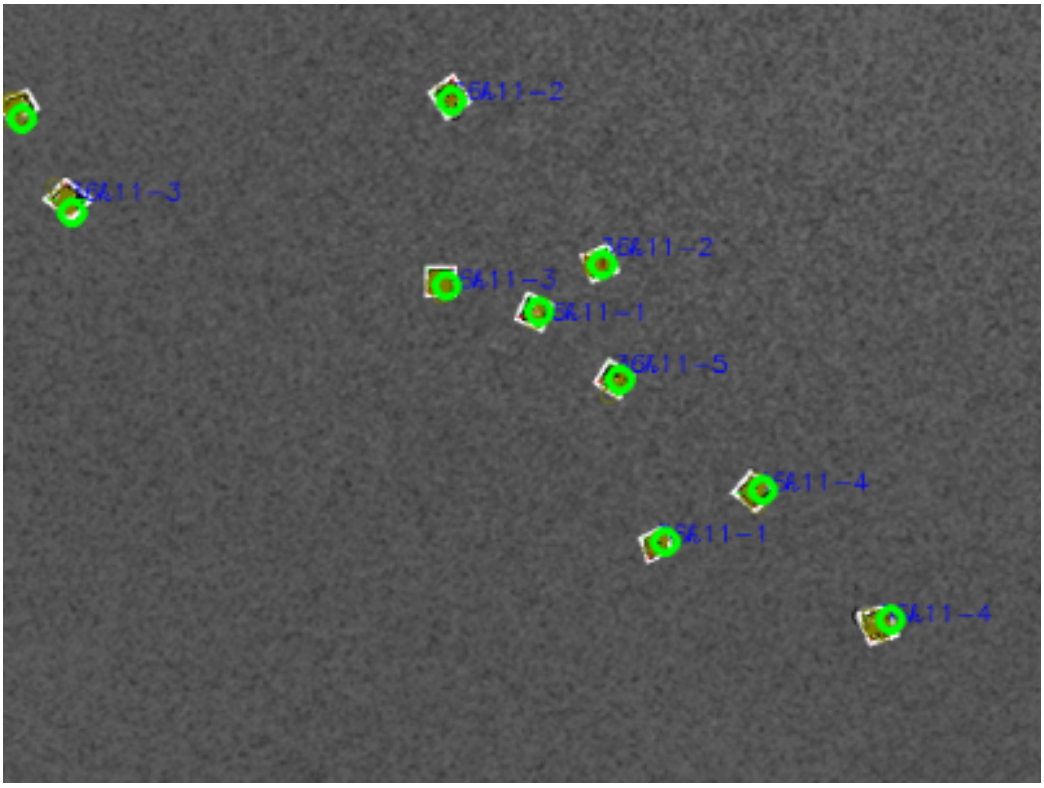
(a) Radial target trajectories (b) Random target trajectories

Figure 5.22: Figures show the trajectories of the targets and EKFs using (5.5).

number of the state vector is increased. In fact, this method was implemented within the PF-EKF framework, but it never worked faster than 0.2 Hz with five targets using 200 particles. Instead, the particle filter approaches utilizes clustering method k-means to identify the number of targets. The k-means algorithm clusters particles based on the positions of the particles and calculates the sum of the variances of all the clusters. Up to five different



(a) Simulation scene



(b) On-board image

Figure 5.23: Figures show the simulation scene and on-board image. The results of the Rao-Blackwellized particle filter are laid over the figures. Although AprilTags have IDs for each pattern, these IDs are not used to classify detections in order to validate the proposed algorithm, which identifies the number of targets just from the distribution.

cluster numbers are used (the number of the measurements $\pm$ 2), and the clusters minimizing the sum of the variance are used to compute the output of the filter. When a sufficient number of particles, defined as 5$\%$ of the total particles, do not belong to a cluster, the cluster is ignored when computing the output. Figures 5.21 and 5.22 show the results of multi-target tracking experiments. Both multiple-hypothesis approach and extended EKF approach explained in (5.5) reasonably tracks multiple targets. One thing found through this experiment is that both methods face difficulty of body attitude estimation. The attitude estimation of markers are used to update the attitude of the body; however, visual measurements are not zero-mean measurements. Even with AprilTags, which are very accurate and robust visual fiducial methods, the attitude estimations are slightly biased. This bias is observed both in Figure 5.21 and 5.22. The impact of the biases is more severe before increasing the measurement error covariance, which is originally tuned for single target tracking. When GPS and magnetometers are functioning healthily, the filter would have minimum benefits of integrating marker attitude into the filter in practice. Figure 5.23 show the simulation scene and on-board image displaying the multi-target tracking experiment. When all the targets are visible most of time, as is the case in Figure 5.21a and 5.22a, there are no visible differences on estimation performance although there is substantial difference in computational cost, which is explained in the following section. When some targets intersect, and visual measurements are temporarily unavailable, the EKF method sometimes unnecessarily increases the number of targets. This is partially because in EKF, the Mahalanobis distance is the unitary criterion to separate targets; on the other hand, the multiple-hypothesis methods offer the variance of the cluster, number of particles on the cluster, and weights of each particle. Also, when targets intersect, EKFs are updated with wrong target as shown in Figure 5.22b.

5.5 Evaluations of EKF, PF-EKF, and RBPF

5.5.1 Computational costs of multiple hypothesis approaches

Computational costs of EKF, PF-EKF, and RBPF are analyzed using image-in-the-loop simulations. In this flight simulator, time step (dt) is set to 0.0001 (10000 Hz), and when the process takes more than the desired time step, it tries to run as fast as possible. The image is processed in a separated thread. When an OpenGL-based scene generator produces a new image, the image is provided for the image processor, in which target detection algorithms run. When all the image processing operations are finished, a scene generator creates the next image. In other words, the update rate of the simulation scene (frames per second, FPS) is mainly determined by how fast the detection algorithms run. Although simulation and image processing run on different threads, they share many parameters, and one thread locks the other when accessing the shared parameters using the technique known as *Mutex locks*. The desired frames per second (FPS) for the scene generator is set to 20 FPS. The results of this experiments are summarized in Table 5.1 and Table 5.2. Figure 5.24 plots the computational costs of all three frameworks. Although the EKF for a single target tracking runs faster than 1500 Hz, it is shown as 1500 Hz for the clarity of the figure. In Figure 5.24, the numbers inside of the parentheses indicate the number of targets tracked in the experiment. For the EKF, the update rates are 4926.1 Hz, 298.5 Hz, and 46.5 Hz for single target, 5 targets, and 10 targets, respectively. As is shown in (5.5), tracking multiple targets in the EKF significantly increases the state space dimension. The standard EKF is by far the fastest method in single target tracking, but as the number of the targets increases, the EKF is slower than the multiple-hypothesis approaches. RBPF is faster than PF-EKF in any category of the experiments when the same number of the particles is used. Note that the data of PF-EKF for multi-target-tracking with 1500 particles could not be obtained. This is because the update rate is too low with such a large number of the particles, and the controller does not work stably. This test is conducted while a vehicle hovers directly over

targets.

Table 5.1: **Computational Cost and Frames per Second (FPS) with Different Particle Numbers and Image Resolutions using PF-EKF**

N	Single target		5 targets		10 targets	
	update rate	FPS	update rate	FPS	update rate	FPS
1500	12.8 Hz	11.4	-	-	-	-
1000	21.1 Hz	11.9	16.6 Hz	5.5	16.7 Hz	5.6
400	80.6 Hz	9.5	46.7 Hz	6.7	42.9 Hz	7.4
200	173.3 Hz	11.7	91.6 Hz	7.6	84.7 Hz	7.6
100	357.1 Hz	11.9	164.7 Hz	7.6	147.7 Hz	7.7
75	469.5 Hz	12.0	207.0 Hz	7.6	174.6 Hz	7.8
50	689.6 Hz	11.8	277.0 Hz	7.5	217.9 Hz	8.1

Table 5.2: **Computational Cost and Frames per Second (FPS) with Different Particle Numbers and Image Resolutions using RBPF**

N	Single target		5 targets		10 targets	
	update rate	FPS	update rate	FPS	update rate	FPS
1500	17.1 Hz	5.8	13.9 Hz	4.6	12.0 Hz	4.7
1000	46.5 Hz	7.2	32.6 Hz	5.2	27.7 Hz	5.3
400	139.7 Hz	10.0	82.6 Hz	6.8	70.9 Hz	7.3
200	362.3 Hz	14.7	153.6 Hz	7.7	132.1 Hz	7.7
100	675.7 Hz	11.9	289.0 Hz	7.8	206.2 Hz	7.7
75	840.3 Hz	11.9	336.7 Hz	7.8	239.8 Hz	7.8
50	1280.4 Hz	11.7	444.4 Hz	7.9	292.4 Hz	7.6

5.5.2 Tracking accuracy while landing

This subsection compares the performance of the three filters in the use of vision-based landing. The landing on a moving target and static target is tested. Figure 5.25 shows the results of one experiment. Here, all three filters are run simultaneously, and the control is based on the true target states so if even one filter fails, the vehicle does not become unstable. Since the last subsection showed that the RBPF is nearly twice as fast as the PF-EKF

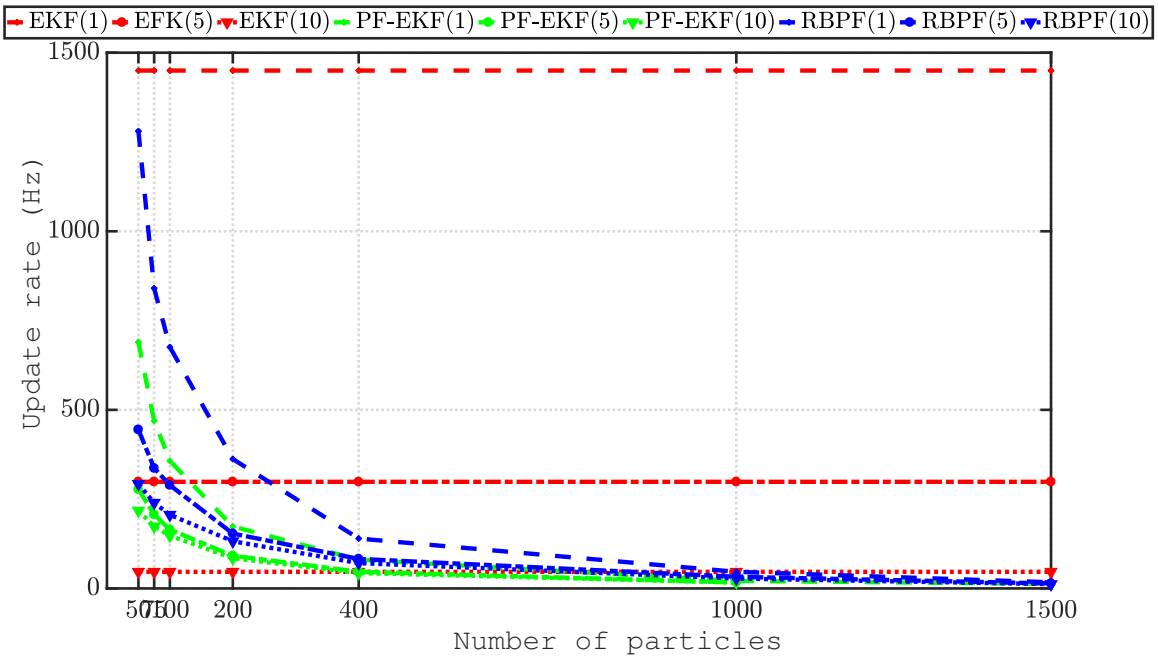


Figure 5.24: Figure summarizes the results of Table 5.1 and Table 5.2 as well as the computational cost of EKF.

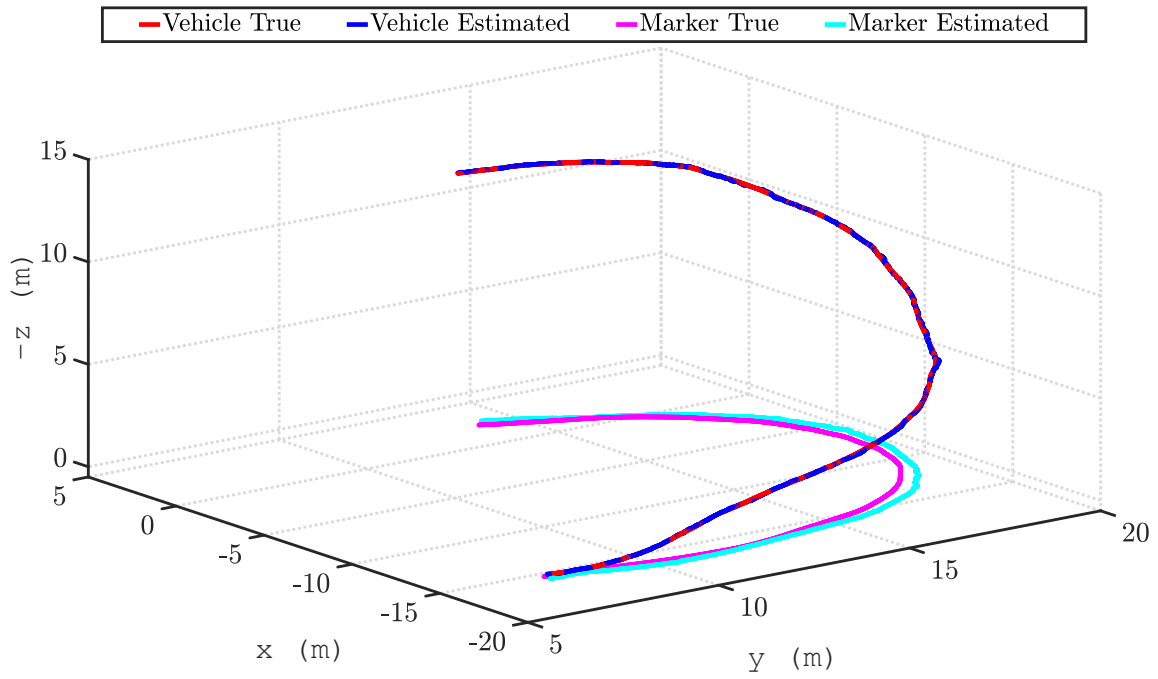
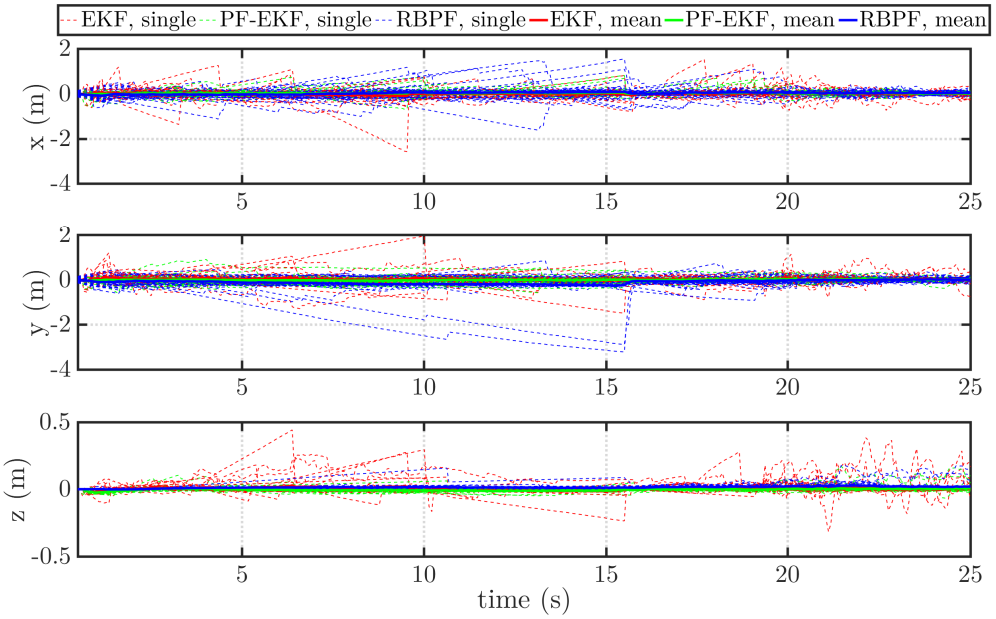


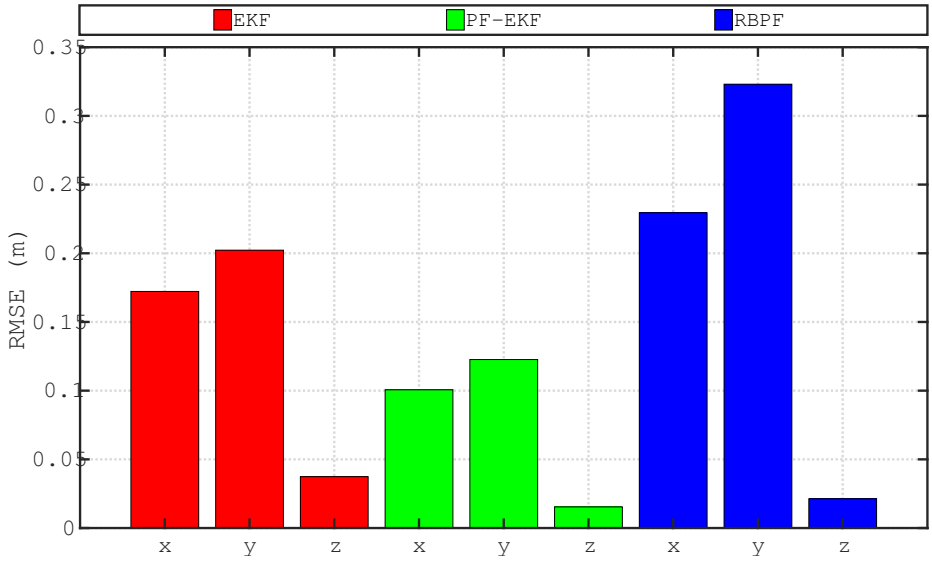
Figure 5.25: Figure shows the true and estimated positions of the vehicle and marker while vision-based landing is conducted. The estimator uses Rao-Blackwellized particle filter for this test.

when the same number of the particles is used, the RBPF utilizes double the number of particles over the RBPF method to fairly compare the estimation performance. Throughout this experiment, 200 particles are used in PF-EKF and 400 particles are used in RBPF. The capability to track multiple target is turned off for EKF, and the measurement outside of the Mahalanobis distance is simply ignored. Figure 5.26 shows the tracking error of 40-runs and root mean square error when landing on a moving target. The mobile target is moving at 8ft/s. Overall, all three filters estimate the position of the marker fairly well, and the RMSEs of lateral directions fall within 0.3 m. As is discussed in Section 5.1, velocity estimation with particle filters leaves many free design parameters whilst the EKF approaches provides a single straightforward solution (Kalman equation). Particles resampled near an existing particle are initialized with the velocity of the existing particle. However, when a particle is resampled near a measurement, there is no good velocity to initialize with a single vision measurement. As a particle filter converges, the output (mean of the particle) becomes more reliable, and the resampling with the velocity of the output makes sense. However, when all the particles are initialized with zero velocity, the mean of the particle also produces the output of the zero velocity. Particles typically initially requires non-zero velocity at initialization. RBPF utilizes the pseudo measurements, which are the position differential, to estimate the velocity. This mitigates the issue of the velocity estimation. However, resampling makes many particles 'jump' very far from one place to another due to false positives and wrong velocity. Only the particles resampled from a correct measurement and subsequent correct measurements produces decent velocity estimation, which would be difficult to make with 400 particles. Because of the problems addressed above, the RBPF-based landing performs least effectively for a moving target. However, RBPF demonstrated the best performance across three configurations for landing on a static target. Since all three filters have very small tracking errors (MRSE of less than 0.1 m), simple comparison of the tracking errors may not show meaningful results. However, while the other filters have some cases that significantly deviated from the ac-

cepted range (Figure 5.27), RBPF maintained the accurate measure. The performance of EKF and PF-EKF are very similar for both static and mobile targets. However, as most obviously seen at 20-25 s in Figure 5.26, the EKF estimation occasionally becomes very rough. At 20-25 s, the landing is the final phase, and marker is highly occluded. In the last minute of touch down, the highly nonlinear multimodal portion detection is used, and this method also includes many false positives. The capability to handle false positives and these complex measurement distributions is only provided to PF-EKF, and the figures show that difference. The condition numbers of the error covariance matrix are also recorded. Figure 5.28 shows the condition numbers of three filters while tracking multiple static AprilTags. When a measurement is highly reliable, as AprilTags are, the accuracy of the estimation of the body states, which typically contributes the smallest singular values, is nearly identical for the three filters. The condition numbers of multiple-hypothesis approaches (PF-EKF and RBPF) are computed using the mean of the particles. The PF-EKF has a slightly smaller estimation error, the condition number of PF-EKF is typically smaller than that of EKF. Since RBPF does not have the marker states, which contribute to the greatest singular value, RBPF typically has the lowest condition number. However, these differences would be negligible from the standpoint of numerical stability.

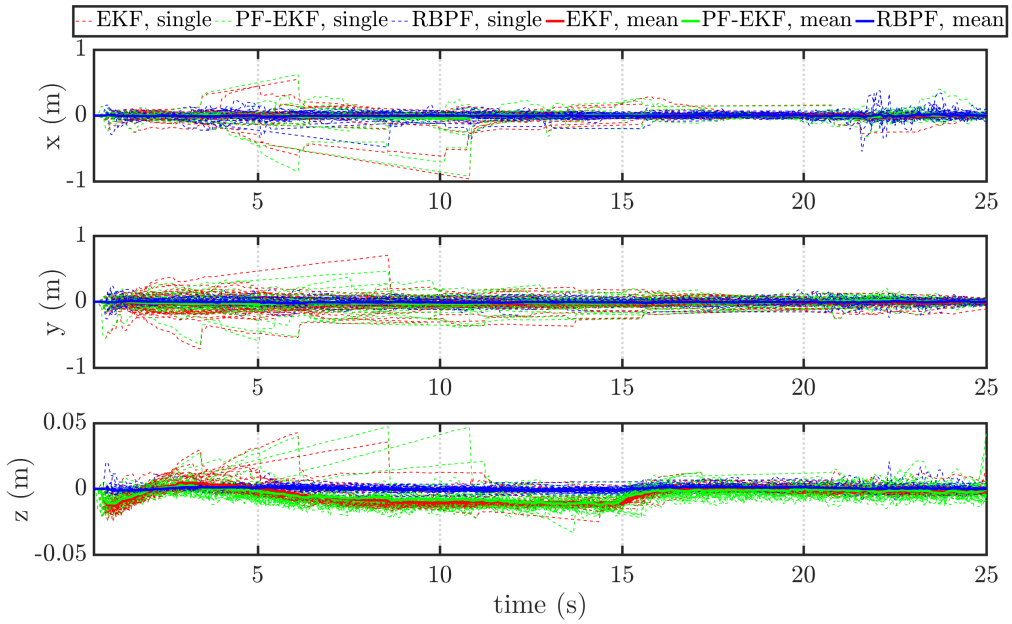


(a)

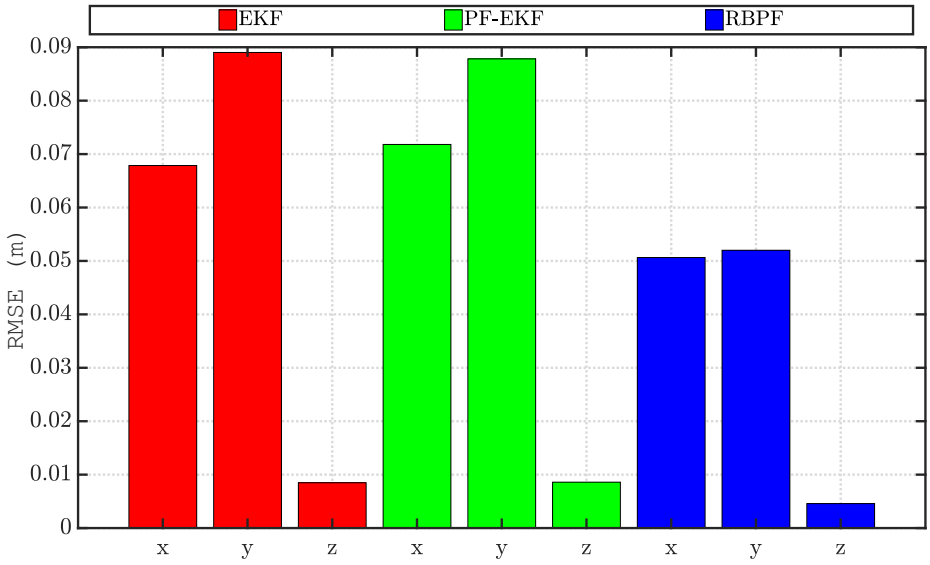


(b)

Figure 5.26: Figures show the estimation error of marker position while vehicle landing on a moving marker. This is the results of 40-run Monte Carlo simulation. The root mean square error (RMSE) are computed for each axis, and the RMSEs of three different filters are compared.



(a)



(b)

Figure 5.27: Figures show the estimation error of marker position while vehicle landing on a static marker. This is the results of 40-run Monte Carlo simulation. The root mean square error (RMSE) are computed for each axis, and the RMSEs of three different filters are compared.

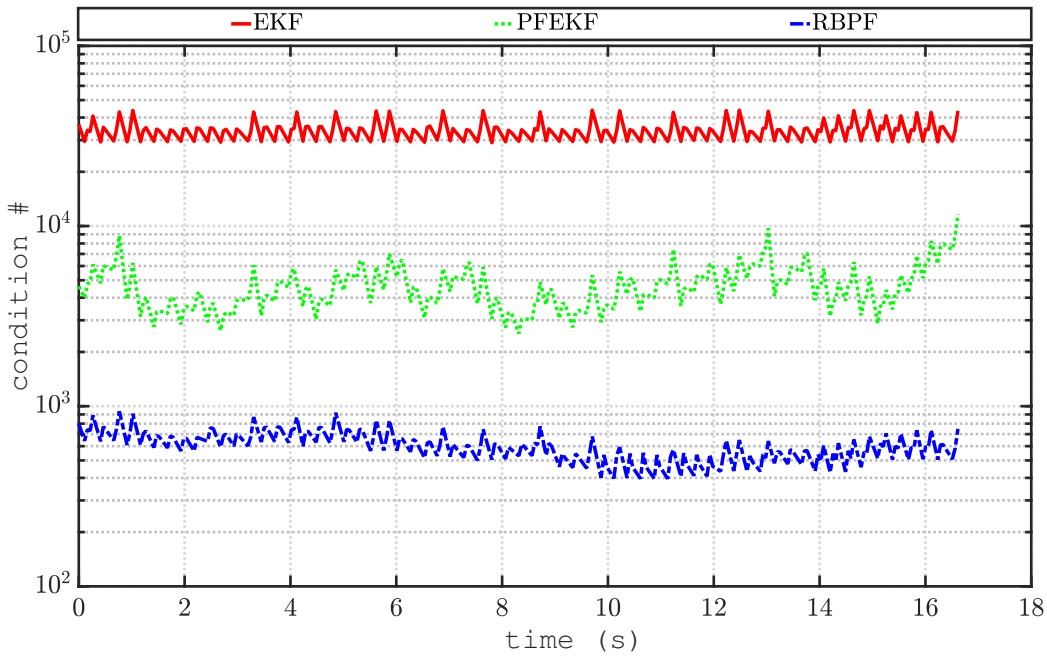


Figure 5.28: Figure shows the condition number of estimated error covariance matrix in EKF, PF-EKF, and RBPF when tracking static AprilTags.

CHAPTER 6
CONCLUSION

This chapter summarizes the conclusions and contributions of this thesis and also identifies recommended future research. This thesis proposes a new vision-based landing architecture to solve an estimation problem with visual measurements. Multiple-hypothesis approaches, specifically the extended Kalman particle filter (PF-EKF) and the Rao-Blackwellized particle filter (RBPF), handle the issues of false positives/negatives, nonlinearity, multi-modality, and non-Gaussianity inherent in vision-based detections. These algorithms are used in the visual SLAM framework, which allows the system to estimate the states of the vehicle and the landing states simultaneously. This way, the navigation solution of the vehicle and the target are not collapsed even when a GPS signal is lost. A new detection algorithm is developed to allow a system to observe a visual marker at various altitudes. This detection algorithm offers a powerful solution to occlusion problems, which are inevitable for vision-based landing. The results from numerical simulations, image-in-the-loop simulations, and the flight tests show that the proposed approach improves the accuracy, scalability, and consistency of the estimation over the existing methods. The results also include the discussion from many practical standpoints: numerical stability, computational cost, initialization, and implementation.

6.1 Contributions

The primary contributions of the thesis are fourfold:

1. Derives vision-aided navigation algorithms within EKF, PF-EKF, and RBPF capable of running in real-time with imagery obtained from a monocular vision camera. These algorithms estimate the states of the vehicle using vision and map the position,

velocity, and attitude of the target utilizing the SLAM framework. The effectiveness of the hybrid methods of EKF and PF for SLAM applications are demonstrated in multiple studies [56], [57], including the well-known FastSMAM approach [58]. The originality of the design proposed in this thesis is that the state vector includes the velocity of the vehicle and targets as well as the sensor biases, which are more appropriate design for fast-dynamics applications such as UAVs. These hybrid methods are reported to be successful particularly in marginalization of vision measurements [44], [59], but to my knowledge, this study is the first to test the marginalization of a full state estimator for both an aircraft and a separate moving platform that includes vision measurements relative to the moving platform. Feasibility of the algorithms is demonstrated with image-in-the-loop simulation and flight test data.

2. Develops a method for detecting a portion of a landmark when a camera is too close to see an entire landmark. The details of the implementation of the method are provided, and the stochastic properties of this method are studied.

3. Proposes a new landing architecture integrating the newly developed detection algorithm in the framework of SLAM. This integration extends the detectable range of the vision system for any known marker and improves the accuracy of the estimation during landing by detecting a target until the last minute of touchdown.

4. Provides numerous test data across three different algorithms (EKF, PF-EKF, and RBPF) to give insights into the advantages and disadvantages of each method. These data display the performance of the algorithms from the perspectives of estimation errors, computational costs, numerical stability, multi-target tracking capability, and robustness to false positives.

6.2 Future work

1. **Extension of Rao-Blackwellization to other filtering techniques**. In this work, Rao-Blackwellization marginalizes the states estimated in the extended Kalman filter (EKF) and the particle filter (PF). However, the Rao-Blackwellization is not limited to this combination, and the other filtering techniques would be able to take an advantage of the marginalization. One promising direction to pursue is to marginalize the states estimated in the unscented Kalman filter (UKF) and PF. Most of the models described in this work can be directly utilized in UKF framework.

2. **Investigation of active estimation**. This thesis solely focuses on the design of estimators; however, by actively controlling a vehicle into a direction that reduces the estimated error covariance matrix, the performance of the estimator would be improved. Considering the fact that many of the recent UAVs are equipped with a controllable gimbal system, it may also be possible to move the camera toward the direction to reduce estimation error.

3. **Evaluation of alternate detection algorithms**. In the present work, the detection algorithms are completely model-based, both in detecting a portion of markers and extracting positions and orientations from the measurements. Considering the recent advancements in deep-learning technology, it may be possible to train a detector that directly extracts the relative position and orientation [128] of a landmark from a portion of a marker. There are multiple frameworks that handle the sequence of images well (e.g. recurrent convolutional neural networks [129]), and they may even be able to extract the relative velocities.

Appendices

APPENDIX A

EXPONENTIAL MAPPING FOR A SMALL ANGLE ROTATION

A matrix exponential of a skew-symmetric matrix can be computed with the Rodrigues formula (2.15), but this formula is numerically unstable when the magnitude of a rotation is small. In that case, an approximated matrix exponential matrix should be used. Consider the first nine terms of the matrix exponential as follows:

$$\begin{aligned}\exp([\rho]_{[\times]}) &= \sum_{k=0}^{\infty} \frac{[\rho]_{\times}{}^{k}}{k!} \\ &\approx I + [\rho]_{\times} + \frac{1}{2}[\rho]_{\times}^{2} + \frac{1}{6}[\rho]_{\times}^{3} + \frac{1}{24}[\rho]_{\times}^{4} ... \\ &+ \frac{1}{120}[\rho]_{\times}^{5} + \frac{1}{720}[\rho]_{\times}^{6} + \frac{1}{5040}[\rho]_{\times}^{7} + \frac{1}{40320}[\rho]_{\times}^{8}\end{aligned} \tag{A.1}$$

Remember the following property of $[\rho]_{\times}$:

$$[\rho]_{\times}^{3} = -\theta^{2}[\rho]_{\times}. \tag{A.2}$$

Then, the approximation can be expressed by using only $[\rho]_{\times}$ and $[\rho]_{\times}^{2}$:

$$[\rho]_{\times}^{4} = -\theta^{2}[\rho]_{\times}^{2}, \tag{A.3}$$

$$[\rho]_{\times}^{5} = -\theta^{2}[\rho]_{\times}^{3} = -\theta^{2}(-\theta^{2}[\rho]_{\times}) = \theta^{4}[\rho]_{\times}, \tag{A.4}$$

$$[\rho]_{\times}^{6} = \theta^{4}[\rho]_{\times}^{2}, \tag{A.5}$$

$$[\rho]_\times^7 = \theta^4[\rho]_\times^3 = \theta^4(-\theta^2[\rho]_\times) = -\theta^6[\rho]_\times, \tag{A.6}$$

$$[\rho]_\times^8 = -\theta^6[\rho]_\times^2. \tag{A.7}$$

Therefore,

$$\exp([\rho]_\times) \approx I + (1 - \frac{1}{6}\theta^2 + \frac{1}{120}\theta^4 - \frac{1}{5040}\theta^6)[\rho]_\times + (\frac{1}{2} - \frac{1}{24}\theta^2 + \frac{1}{720}\theta^4 - \frac{1}{40320}\theta^6)[\rho]_\times^2. \tag{A.8}$$

This formula is used when θ is small. The threshold is tuned considering the computing environment.

APPENDIX B

IMU PLACEMENT

In Section 3, it is assumed that the IMU is placed exactly on the C.G. location of the body, which is also the origin of the body coordinated. However, in a real situation, encountering to a difficulty in placing an IMU on C.G. leads to the necessity of compensating for the inertial acceleration, a.k.a. pseudo acceleration. In this Appendix, the effects of the offset of the IMU is evaluated, and the way to compensate for the inertial acceleration is introduced.

First of all, when it comes to the effects of the offset of the IMU (denoted $p^b_{b\to\mathrm{imu}}$), the contributor to the error is an accelerometer, not a gyroscope. That is because a measurement of angle, angular velocity, and angular acceleration is the same as the one of the body as long as the IMU is fixed to the body. However, the measurement of the acceleration on the rotating frame introduces the effects of the offset:

$$s^{\mathrm{imu}}_{nb} = s^b_{nb} + \dot{w}^b_{nb} \times p^b_{b\to\mathrm{imu}} + 2w^b_{nb} \times v^b_{b\to imu} + w^b_{nb} \times (w^b_{nb} \times p^b_{b\to\mathrm{imu}}), \tag{B.1}$$

The first term is the specific acceleration of the body (C.G. location) relative to the local north-east-down (NED) coordinate, and the second term is called Euler acceleration, or tangential acceleration responsible for rotational acceleration of moving frame. The third term is called Coriolis acceleration, and the fourth term is called the centripetal acceleration [130]. Here, the IMU is assumed to be fixed to the body, which is not flexible ($v^b_{b\to imu} = \dot{p}^b_{b\to imu} = 0$); thus, the compensation model is reduced to the equation as follows:

$$s^b_{nb} = s^{\mathrm{imu}}_{nb} - \dot{w}^b_{nb} \times p^b_{b\to\mathrm{imu}} - w^b_{nb} \times (w^b_{nb} \times p^b_{b\to\mathrm{imu}}). \tag{B.2}$$

In the EKF used in this thesis, the angular velocity is not added to the state vector, and the best estimation of the angular velocity is the raw gyroscope measurement compensated for

the estimated bias. The estimated angular velocity is expressed as follows:

$$\hat{w}^b_{nb} = z_{gyro} - \hat{b}_{\text{gyro}}. \tag{B.3}$$

One way to obtain an angular acceleration is through back-differencing the estimated angular velocity:

$$\tilde{\dot{w}}^b_{nb}(t) = \frac{\hat{w}^b_{nb}(t) - \hat{w}^b_{nb}(t-dt)}{dt}. \tag{B.4}$$

However, the angular acceleration obtained in (B.4) is a rough measurement. Before used for the compensation of the inertial acceleration, the measured angular acceleration is filtered in a first-order filter as follows:

$$\frac{d}{dt}\hat{\dot{w}}^b_{nb} = \{\frac{\tilde{w}^b_{nb}(t) - \tilde{w}^b_{nb}(t-\Delta t)}{\Delta t} - \hat{\dot{w}}^b_{nb}\}\frac{1}{\tau_{\dot{w}}}. \tag{B.5}$$

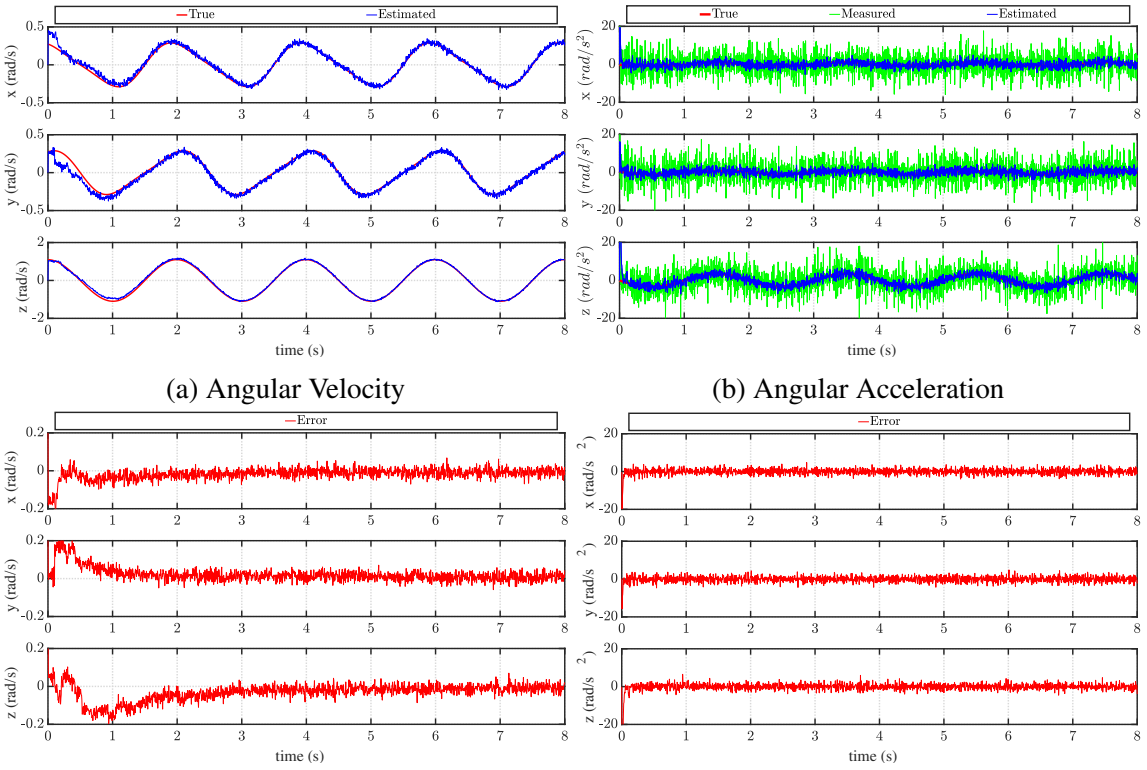
The results of the estimation using (B.3), (B.4) and (B.5) are shown in Figure B.1. When an IMU is not placed on the C.G. of the body, the measurement of accelerometer (3.6) is replaced by the one consisting of inertial accelerations is shown as below:

$$\begin{aligned} z_{\text{acc}} &= s^b_{nb} + \dot{w}^b_{nb} \times p^b_{b\to\text{imu}} + w^b_{nb} \times (w^b_{nb} \times p^b_{b\to\text{imu}}) + b_{\text{acc}} + \sigma_{\text{acc}}\nu \\ &= R^b_n(a^n_{nb} - g^n) + \dot{w}^b_{nb} \times p^b_{b\to\text{imu}} + w^b_{nb} \times (w^b_{nb} \times p^b_{b\to\text{imu}}) + b_{\text{acc}} + \sigma_{\text{acc}}\nu, \end{aligned} \tag{B.6}$$

and the estimated specific acceleration of the body (3.11) is modified to account for the inertial accelerations as follows:

$$\hat{s}^b_{nb} = z_{\text{acc}} - \hat{\dot{w}}^b_{nb} \times p^b_{b\to\text{imu}} - \hat{w}^b_{nb} \times (\hat{w}^b_{nb} \times p^b_{b\to\text{imu}}) - \hat{b}_{\text{acc}}. \tag{B.7}$$

When $p^b_{b\to\text{imu}} = 0$, (B.6) and (B.7) are equivalent to (3.6) and (3.11). Figure B.2 shows the results of the experiments demonstrating the effects of the IMU locations. When an IMU is placed on a non-C.G. location, the offset is $p^b_{b\to\text{imu}} = \begin{bmatrix} 0.7 & 0.6 & 0.5 \end{bmatrix}^T (m)$. Here,



(a) Angular Velocity

(b) Angular Acceleration

(c) Estimation Error of Angular Velocity

(d) Estimation Error of Angular Acceleration

Figure B.1: Figures show true/estimated angular velocity and acceleration. The estimation errors are also shown.

the synthetic trajectories of the position and Euler angle are generated as sinusoidal curves, and the Euler angle acceleration is analytically derived. Note that the angular velocity of the body is calculated from the rate of the Euler angle as follows:

$$w_{nb}^{b} = I_3 \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_2^b \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_2^b R_1^2 \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}, \tag{B.8}$$

where I_3 is a 3x3 identity matrix,

$$R_2^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{B.9}$$

and

$$R_1^2 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}. \tag{B.10}$$

The 1 and 2 frames are the intermediate frames obtained by rotating the roll and pitch angles from the body coordinate, respectively. Let Φ denote the Euler angle $\Phi_{bn} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$. For the further discussion, the conversion is rewritten as follows:

$$\begin{aligned} w_{nb}^b &= \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \dot{\Phi}_{bn} \\ &= R_{\dot{\Phi}}^w \dot{\Phi}_{bn}. \end{aligned} \tag{B.11}$$

Therefore, the angular acceleration is computed as follows:

$$\dot{w}_{nb}^b = \frac{d}{dt} w_{nb}^b = \frac{dR_{\dot{\Phi}}^w}{dt} \dot{\Phi}_{bn} + R_{\dot{\Phi}}^w \frac{d\dot{\Phi}_{bn}}{dt}. \tag{B.12}$$

Note that $\frac{dR_{\dot{\Phi}}^w}{dt} = \dot{R}_{\dot{\Phi}}^w$ cannot be computed as $\dot{R} = R[w]_\times$ because $R_{\dot{\Phi}}^w$ is not a proper SO(3) matrix. Instead, the derivative needs to be computed simply by the matrix-by-scalar method, which means to compute the tangent matrix as follows:

$$\frac{d}{dt} R_{\dot{\Phi}}^w = \begin{bmatrix} \frac{dr_{00}}{dt} & \frac{dr_{01}}{dt} & \frac{dr_{02}}{dt} \\ \frac{dr_{10}}{dt} & \frac{dr_{11}}{dt} & \frac{dr_{12}}{dt} \\ \frac{dr_{20}}{dt} & \frac{dr_{21}}{dt} & \frac{dr_{22}}{dt} \end{bmatrix}. \tag{B.13}$$

The chain rule $\frac{dy}{dt} = \frac{dy}{dx}\frac{dx}{dt}$ and the product rule $\frac{d}{dt}x \cdot y = \frac{d}{dt}x \cdot y + x \cdot \frac{d}{dt}y$ apply here. Only

the non-zero elements are shown below:

$$\dot{R}^w_{\dot{\Phi}}(1,3) = -\cos(\theta)\dot{\theta}, \tag{B.14}$$

$$\dot{R}^w_{\dot{\Phi}}(2,2) = -\sin(\phi)\dot{\phi}, \tag{B.15}$$

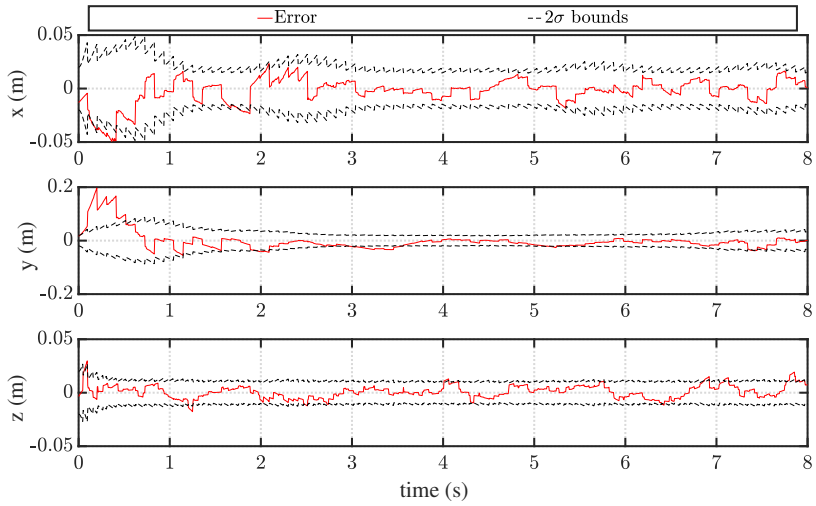
$$\begin{aligned} \dot{R}^w_{\dot{\Phi}}(2,3) &= (\frac{d}{dt}\sin\phi)\cos\theta + \sin\phi(\frac{d}{dt}\cos\theta) \\ &= (\dot{\phi}\cos\phi)\cos\theta + \sin\phi\{\dot{\theta}(-\sin\theta)\} \\ &= \dot{\phi}\cos\phi\cos\theta - \dot{\theta}\sin\phi\sin\theta, \end{aligned} \tag{B.16}$$

$$\dot{R}^w_{\dot{\Phi}}(3,2) = -\cos(\phi)\dot{\phi}, \tag{B.17}$$

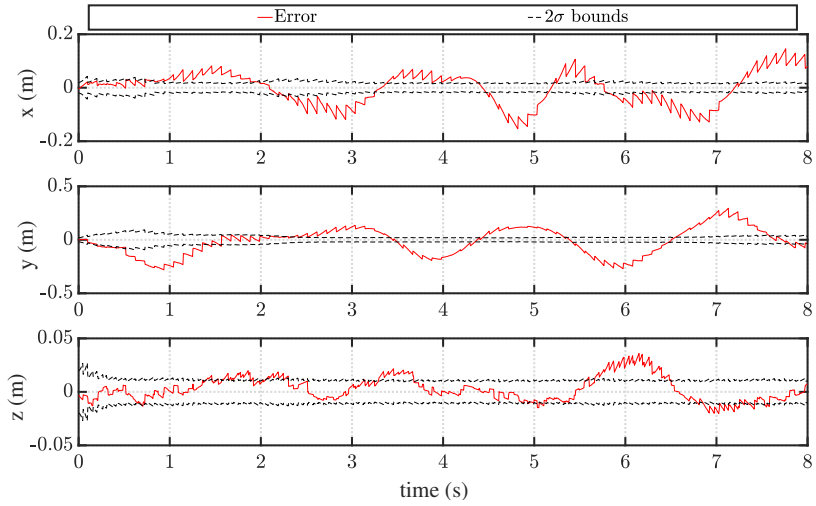
$$\begin{aligned} \dot{R}^w_{\dot{\Phi}}(3,3) &= (\frac{d}{dt}\cos\phi)\cos\theta + \cos\phi(\frac{d}{dt}\cos\theta) \\ &= \{\dot{\phi}(-\sin\phi)\}\cos\theta + \cos\phi\{\dot{\theta}(-\sin\theta)\} \\ &= -\dot{\phi}\sin\phi\cos\theta - \dot{\theta}\cos\phi\sin\theta, \end{aligned} \tag{B.18}$$

and the final results are the following:

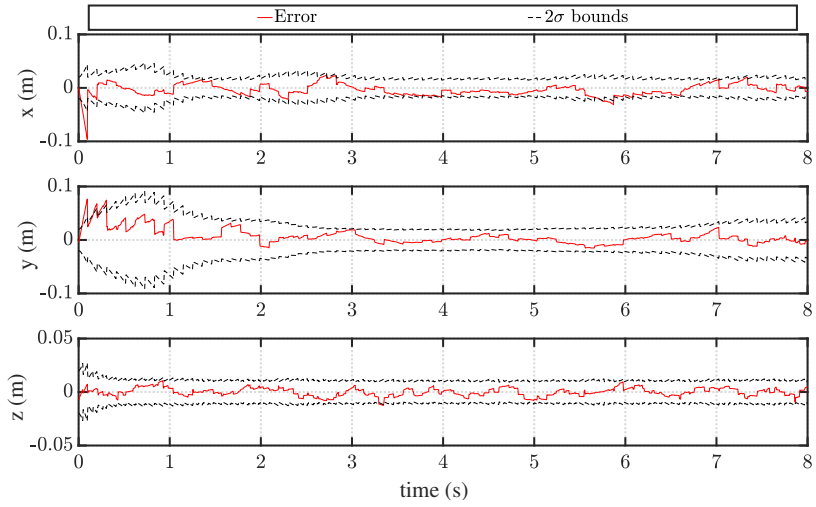
$$\frac{d}{dt}R^w_{\dot{\Phi}} = \begin{bmatrix} \frac{dr_{00}}{dt} & \frac{dr_{01}}{dt} & \frac{dr_{02}}{dt} \\ \frac{dr_{10}}{dt} & \frac{dr_{11}}{dt} & \frac{dr_{12}}{dt} \\ \frac{dr_{20}}{dt} & \frac{dr_{21}}{dt} & \frac{dr_{22}}{dt} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\cos(\theta)\dot{\theta} \\ 0 & -\sin(\phi)\dot{\phi} & \dot{\phi}\cos\phi\cos\theta - \dot{\theta}\sin\phi\sin\theta \\ 0 & -\cos(\phi)\dot{\phi} & -\dot{\phi}\sin\phi\cos\theta - \dot{\theta}\cos\phi\sin\theta \end{bmatrix}. \tag{B.19}$$



(a)



(b)



(c)

Figure B.2: IMU is on C.G. in (a). IMU is on non-C.G. in (b), and EKF assumes the IMU is on C.G. In (c), IMU is on non-C.G., and EKF compensates for the inertial accelerations.

APPENDIX C

VALIDATION OF STANDARD EKF

Without vision measurements, the estimator is a naive EKF fusing the sensor measurements from IMU, GPS, barometric pressure sensor, magnetometer, and more. This EKF is validated through a Monte Carlo simulation without images in the loop.

C.1 Statistics of Kalman innovations

One important technique to examine the implementation of a Kalman filter is to use the knowledge of the statistics of the innovation, or measurement residuals. The innovation is the difference between a measurement and an estimated output, which, in a linear system, is defined as follows:

$$\epsilon_k = z_k - y_k = z_k - H_k \hat{x}_k^-. \tag{C.1}$$

The innovation is a zero-mean, white, stochastic process with a known covariance. The details of the stochastic properties of the innovation are described in Chapter 10.1 of [79]. The covariance of the innovation is derived as follows:

$$\begin{aligned} E[\epsilon_k \epsilon_k^T] &= E[(z_k - H_k \hat{x}_k^-)(z_k - H_k \hat{x}_k^-)^T] \\ &= E[\{H_k(x_k - \hat{x}_k^-) + \sigma\nu_k\}\{H_k(x_k - \hat{x}_k^-)^T + \sigma\nu_k\}^T] \\ &= H_k E[e_k^- e_k^{-T}] H_k^T + E[\sigma_k \sigma_k^T] \ (\because e_k^- = x_k - \hat{x}_k^-) \\ &= H_k P_k^- H_k^T + \Sigma_k = S. \end{aligned} \tag{C.2}$$

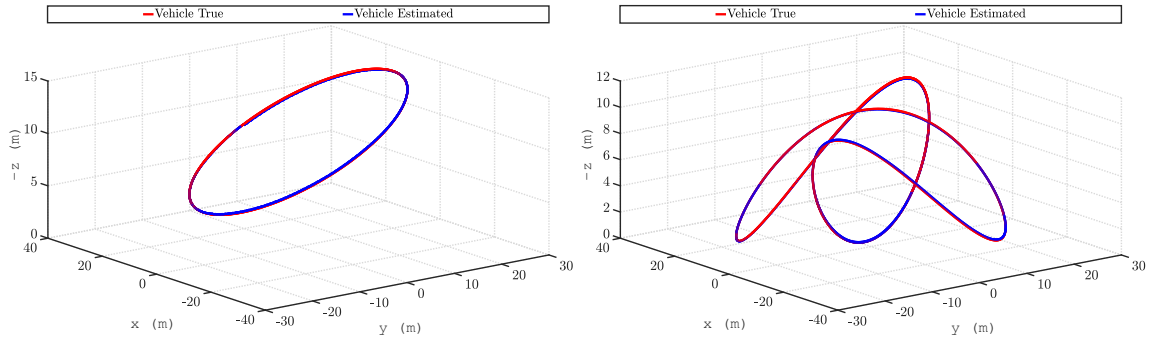
This S matrix is computed in the process of the mean update of Kalman filters as shown in (3.50).

C.2 Test trajectories

In order for the Monte Carlo simulation to be validated, the Kalman filter needs to be tested with various initial conditions and different parameters. The validation is conducted with the synthetic trajectory of either circle or Lissajous. Figure C.1 shows examples of the trajectories used for the validation. Both the circle and Lissajous trajectories are generated as a combination of sinusoidal functions; therefore, the derivatives are easily accessible analytically. The 3-D position and Euler angle are generated by a sinusoidal function:

$$f(A, B, C, D, t) = A\sin(Bt + C) + D, \tag{C.3}$$

and the circle and the Lissajous trajectories can be generated by the different combinations of the parameters A, B, C, and D.



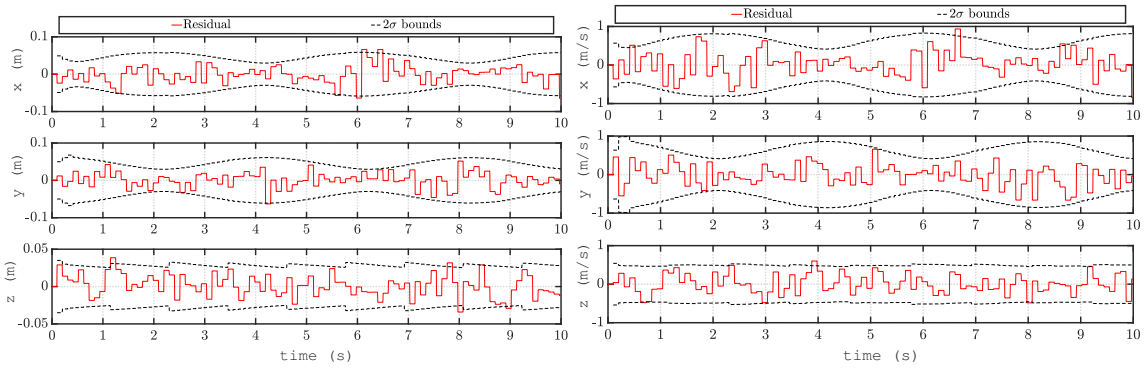
(a) Circle (b) Lissajous

Figure C.1: Figures show examples of the trajectories used for the Monte Carlo simulation.

C.3 Simulation results

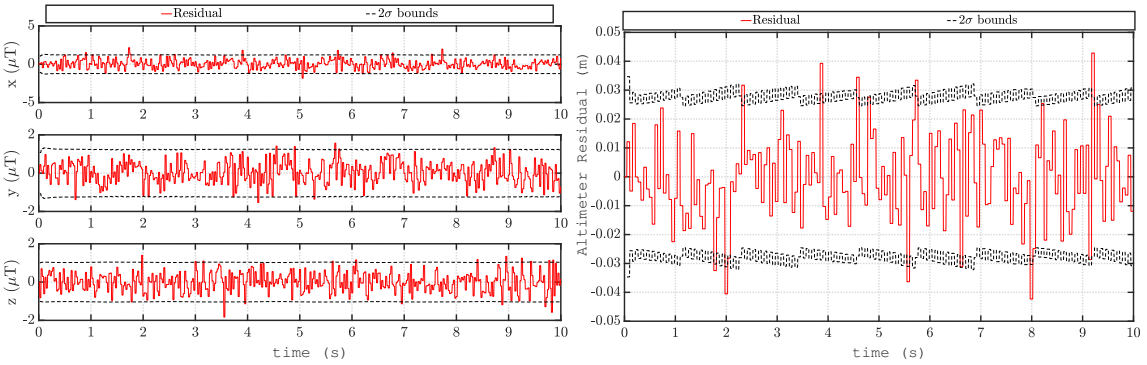
Figure C.2 and C.3 display some examples of the time history of the innovations and their two sigma bounds. Figure C.4 shows the results of the chi square tests produced from the Monte Carlo simulations with the 20,000 samples. Figure C.5 is an example of the time history of the estimated error and their two sigma bounds computed from the estimated error covariance matrix P. Figure C.6 shows the results of the Monte Carlo simulation,

and Figure C.7 shows the chi square tests. Part of the content of this Appendix is published in [131].



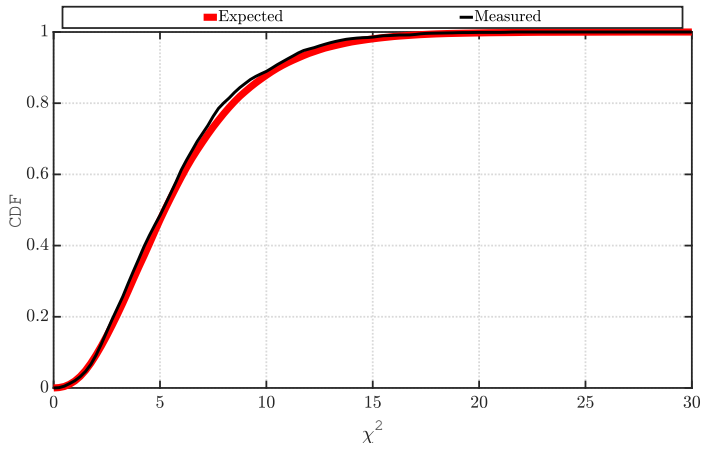
(a) GPS position residuals (b) GPS velocity residuals

Figure C.2: Measurement residuals of the GPS positions (left) and GPS velocity (right). Figures also show their two sigma bounds computed from the estimated *a priori* error covariance matrix. The GPS measurements are available at 10 Hz.

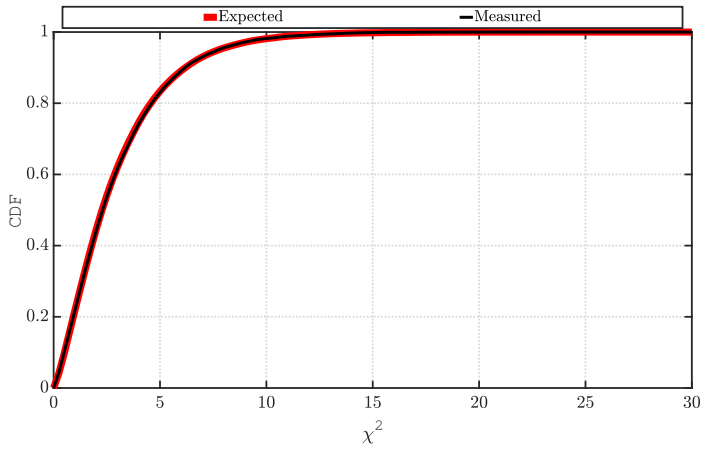


(a) Magnetometer residuals (b) Barometer residuals

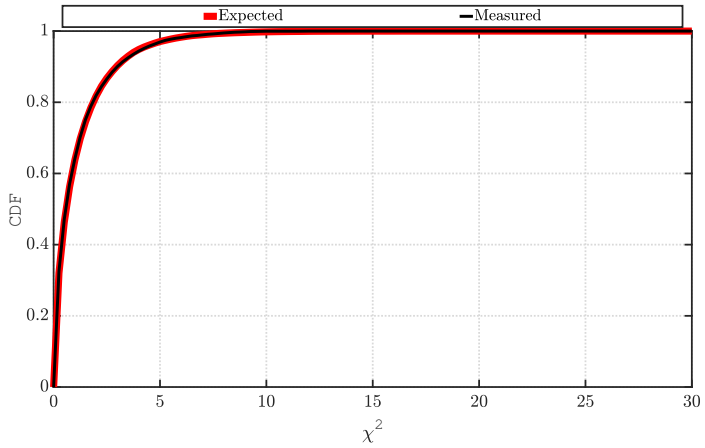
Figure C.3: Measurement residuals of the magnetometer (left) and the barometer (right). Figures also show their two sigma bounds computed from the estimated *a priori* error covariance matrix. The magnetometer measurements are available at 50 Hz, and the ones of barometer are available at 20 Hz.



(a) GPS

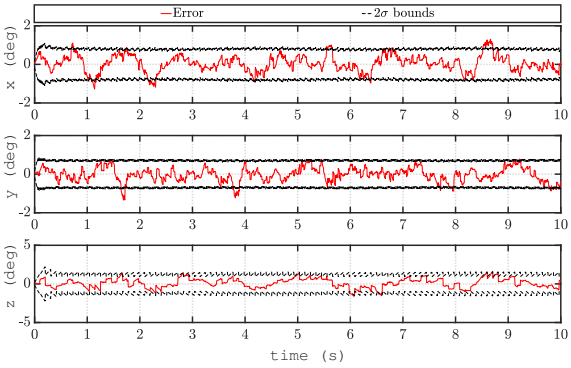


(b) Magnetometer

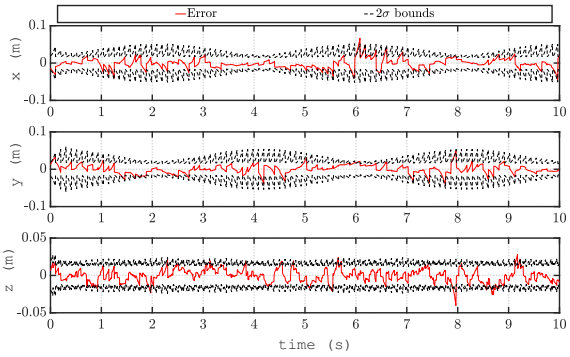


(c) Barometer

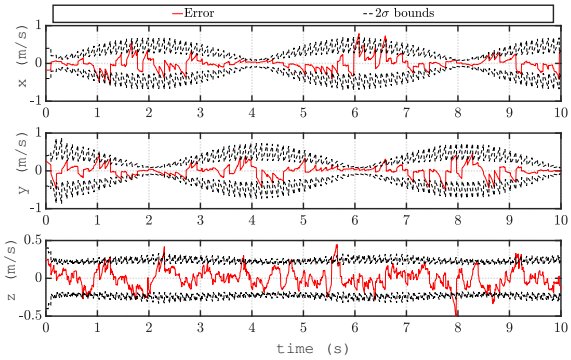
Figure C.4: The results of the χ^2 tests of the innovations. The dimensions of the measurements are 6-DOF, 3-DOF, and 1-DOF for the GPS, the magnetometer, and the barometer, respectively. The results were produced using a Monte Carlo simulation technique, and the number of the samples used to produce the results is 20,000. All three sensors used to update the estimated states show a great fit for their expected stochastic property.



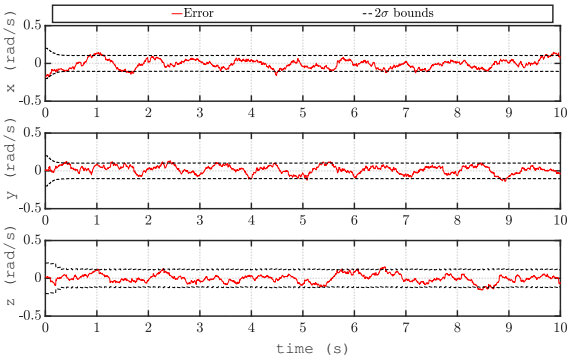
(a) Euler Angle



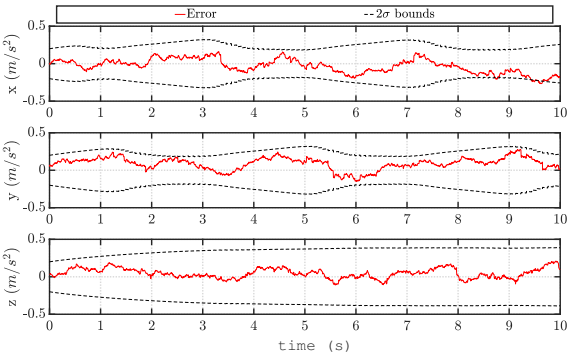
(b) Position



(c) Velocity

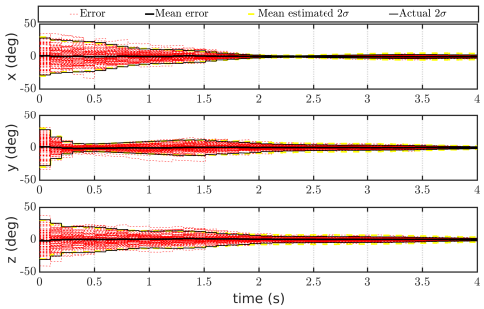


(d) Gyro Bias

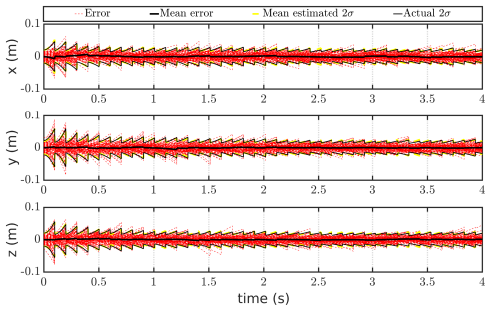


(e) Accelerometer Bias

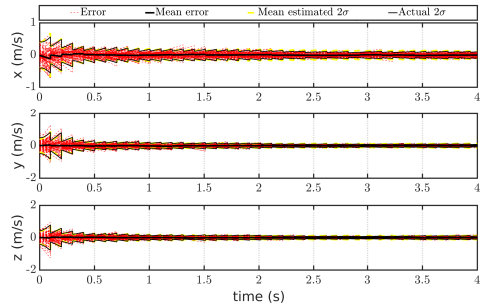
Figure C.5: Examples of the simulation results showing the estimation errors and their 2σ bounds.



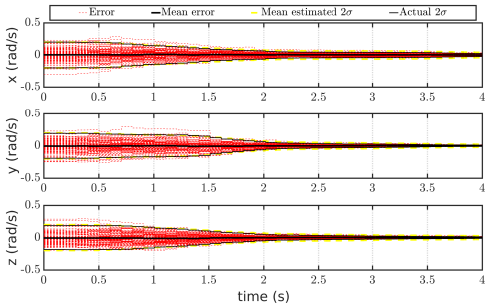
(a) Euler Angle



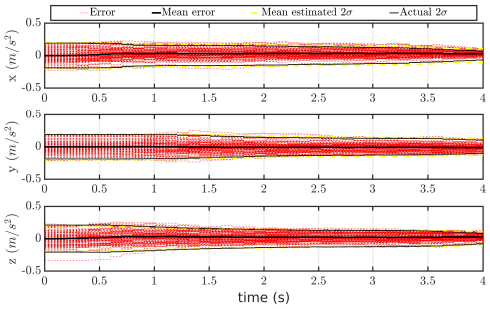
(b) Position



(c) Velocity

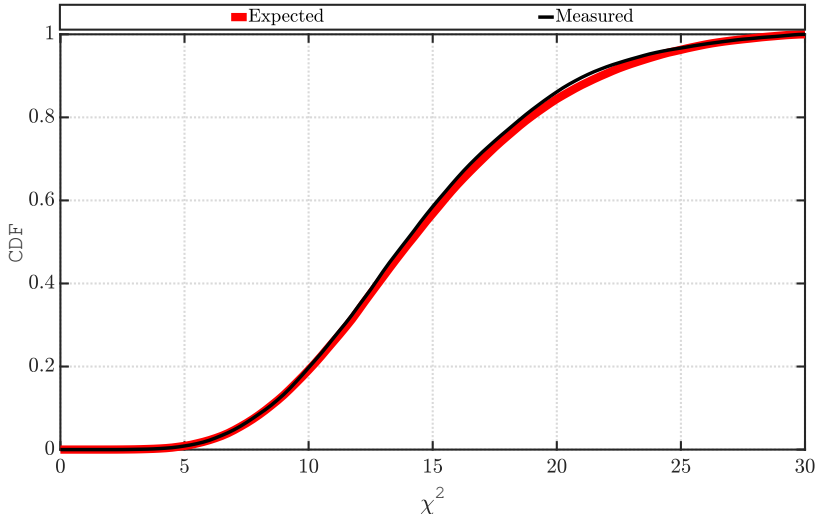


(d) Gyro Bias

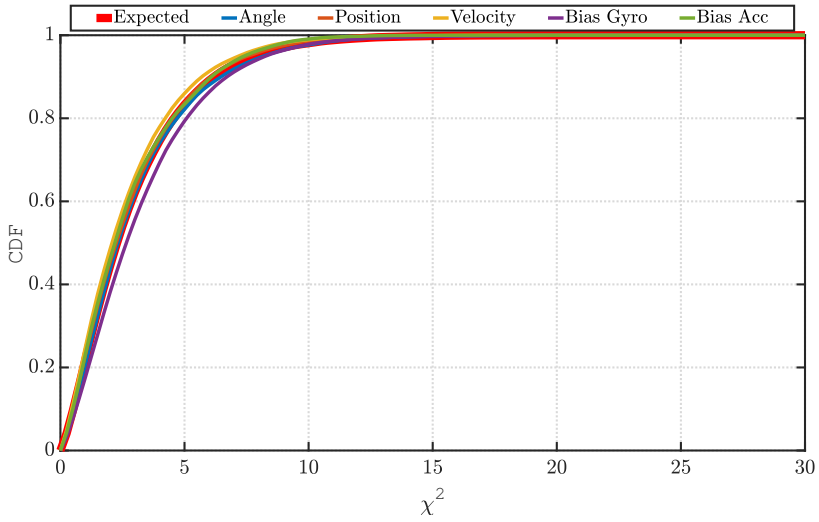


(e) Accelerometer Bias

Figure C.6: Figure shows the results of a Monte Carlo simulation. Each red line represents an estimation error. The solid black line shows the mean error of the entire runs. Estimated 2σ is computed at each run from the error covariance matrix, and the mean of them is displayed on the figure as a yellow line. The 2σ of the entire runs is also computed and shows a great fit to the estimated value.



(a) Entire 15 states



(b) Each 3-vector

Figure C.7: The results of the χ^2 tests of the estimated errors. The left shows the results with the entire 15-states, and the right shows the ones with the each 3-vector.

APPENDIX D

MAGNETOMETER AS 1-AXIS SENSOR

The measurements of a magnetometer is highly biased, non-zero-mean, and non-Gaussian. When the local magnetic disturbance is known, the magnetometer can be used as a 3-axis sensor; however, it is not always the case. When the magnetometer measurements are not particularly reliable, using a magnetometer as an 1-axis sensor is another option. The magnetic field measured by the magnetometer is expressed in a body-fixed frame. Resolve the measurement in a local north-east-down (NED) frame and let $\tilde{B}^n$ denote the measured magnetic field as follows:

$$\tilde{B}^n = \hat{R}_b^n z_{\text{mag}}, \tag{D.1}$$

The measured magnetic north ($\tilde{N}_{\text{mag}}$) is computed as following:

$$\tilde{N}_{\text{mag}} = \text{atan2}(\tilde{B}_y^n, \tilde{B}_x^n). \tag{D.2}$$

In this model, the measurement residual for the Kalman update is

$$\epsilon_{\text{mag},1D} = \tilde{N}_{\text{mag}} - N_{\text{mag}}, \tag{D.3}$$

where the true magnetic north N_{mag} is computed from the known local magnetic field B_n as follows:

$$N_{\text{mag}} = \text{atan2}(B_y^n, B_x^n). \tag{D.4}$$

The axis angle is the same as the Euler angle when the rotation from an NED to body frames consists of only one rotation around one of the axes of the NED frame. For example, the Euler angle $\Phi = \begin{bmatrix} \pi/4 & 0 & 0 \end{bmatrix}^T$, which is the 45 degrees of bank angle, is equivalent to the

axis angle $\rho = \begin{bmatrix} \pi/4 & 0 & 0 \end{bmatrix}^T$, and $\Phi = \begin{bmatrix} 0 & 0 & \pi \end{bmatrix}^T$ is equal to $\rho = \begin{bmatrix} 0 & 0 & \pi \end{bmatrix}^T$. When a magnetometer is used as an 1-axis sensor, the output is treat equivalently both in Euler and axis angles; therefore, the output matrix corresponding to (3.1) is computed as follows:

$$H_{\text{mag},1D} = \frac{\partial y}{\partial \hat{x}} \tag{D.5}$$

$$= \begin{bmatrix} \frac{\partial \tilde{B}^n}{\partial \hat{\rho}_b^n}(3,:) & 0_{1\times 12} \end{bmatrix} \tag{D.6}$$

$$= \begin{bmatrix} -\hat{R}_b^n(3,:) & 0_{1\times 12} \end{bmatrix}. \tag{D.7}$$

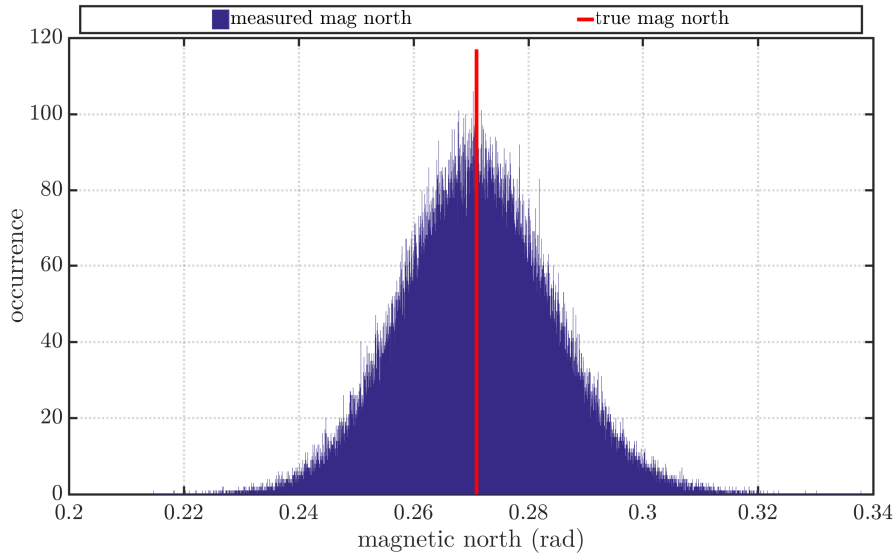
The measurement error covariance of magnetometers (Σ_{mag}) is expressed in the unit of magnetic field. The measurement error covariance of (D.2) is approximated using the Jacobian matrix. The uncertainty of the magnetometer measurement is propagated using the following Jacobian matrix:

$$J_{\text{mag}} = \begin{bmatrix} \frac{\partial \tilde{N}_{\text{mag}}}{\partial \tilde{B}_x^n} & \frac{\partial \tilde{N}_{\text{mag}}}{\partial \tilde{B}_y^n} \end{bmatrix} = \begin{bmatrix} -\frac{y}{x^2\left(1+(\frac{y}{x})^2\right)} & \frac{1}{x(1+\left(\frac{y}{x}\right)^2)} \end{bmatrix}. \tag{D.8}$$

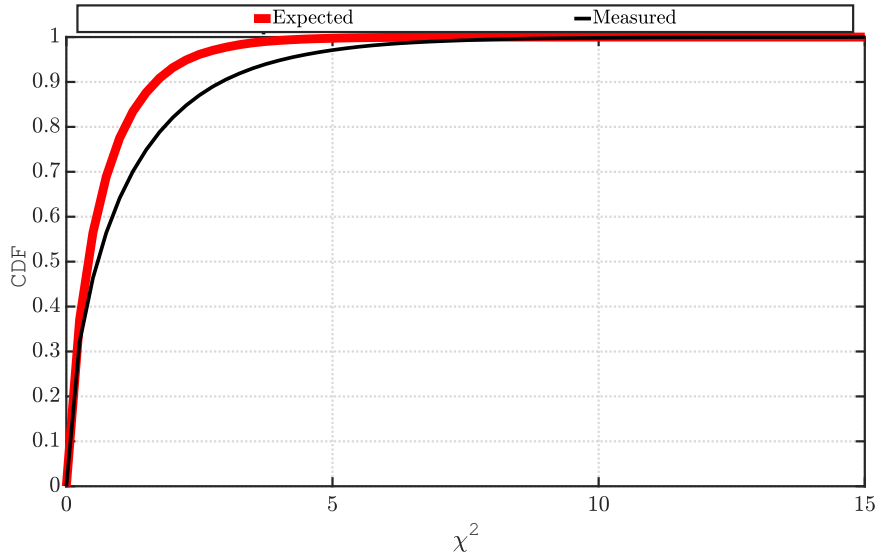
Note $\frac{\partial}{\partial t}\tan^{-1}(t) = \frac{1}{1+t^2}$. The measurement error covariance of the magnetic north is expressed as follows:

$$\Sigma_{\text{mag,1d}} = J\Sigma_{\text{mag}}J^T. \tag{D.9}$$

Since (D.9) is an approximation, the statistics of the 1-axis magnetometer measurements become slightly optimistic (i.e. the estimated variance of the measurement is lower than the actual variance). It is important to make additive noises to estimator to make the estimator resilient to the mismatch of the statistics; in fact, when this 1-axis approach is used, the magnetometer may not be a reliable sensor, and a very large measurement error covariance matrix is typically provided to estimators. The 1-axis approach is typically used when the local magnetic disturbance is unknown. This disturbance is modeled as B_{dist}^b in (3.8). Figure D.2 shows the estimation error of vehicle angles using a magnetometer as 1-D and



(a) Measurement Distribution

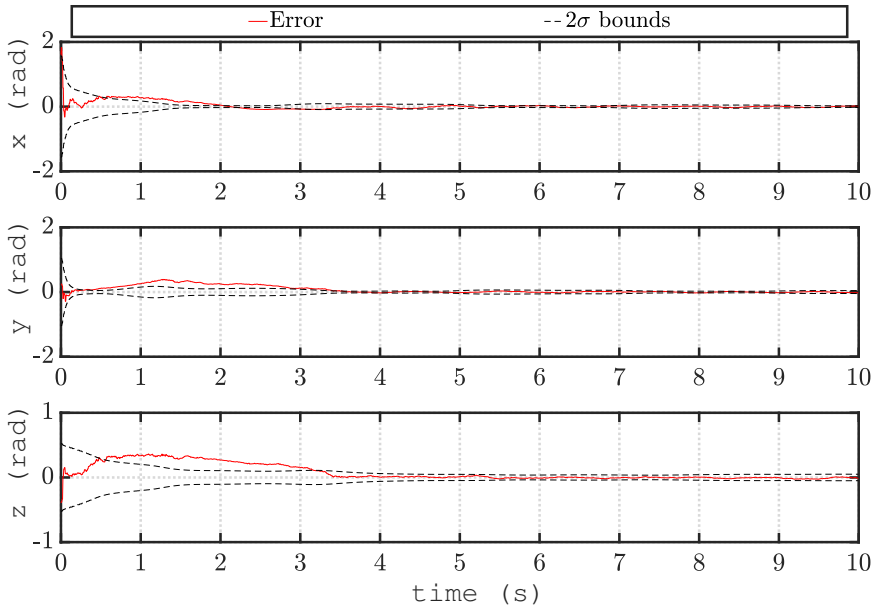


(b) χ^2 of 1-D magnetometer measurement

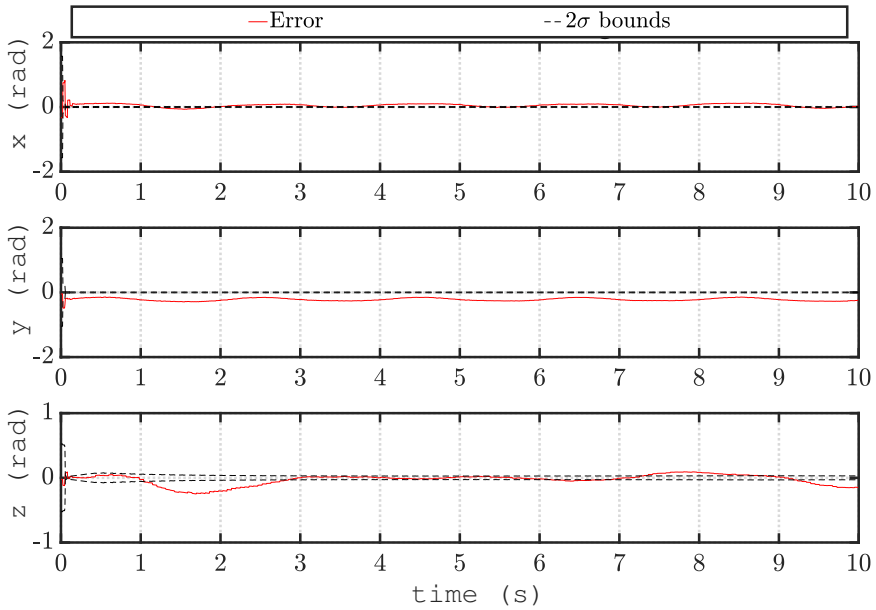
Figure D.1: Figures show the statistics of the one-dimensional magnetometer measurements.

3-D sensors. In this test, $B^b_{\text{dist}} = \begin{bmatrix} 0.01 & 0.02 & 0.03 \end{bmatrix}$ [mT] is added as an unknown local disturbance. The results of both 1-D and 3-D updates do not appropriately fit in the 2 sigma bounds; however, the results with 3-D measurements display a notable bias in the vehicle tilts. The bias in the vehicle tilts (x and y axes) misaligns the estimated gravitational vector; thus, the propagation of the position and the velocity are significantly affected by

this estimation errors. Figure D.3 shows the results of the position estimation. This test is identical to the one conducted to create Figure D.2. The bias of the attitude estimation most significantly affects the position estimation. This 1-D sensor approach is used for flight tests, but in simulation, since there is no unknown local disturbance, the 3-D sensor approach is used.

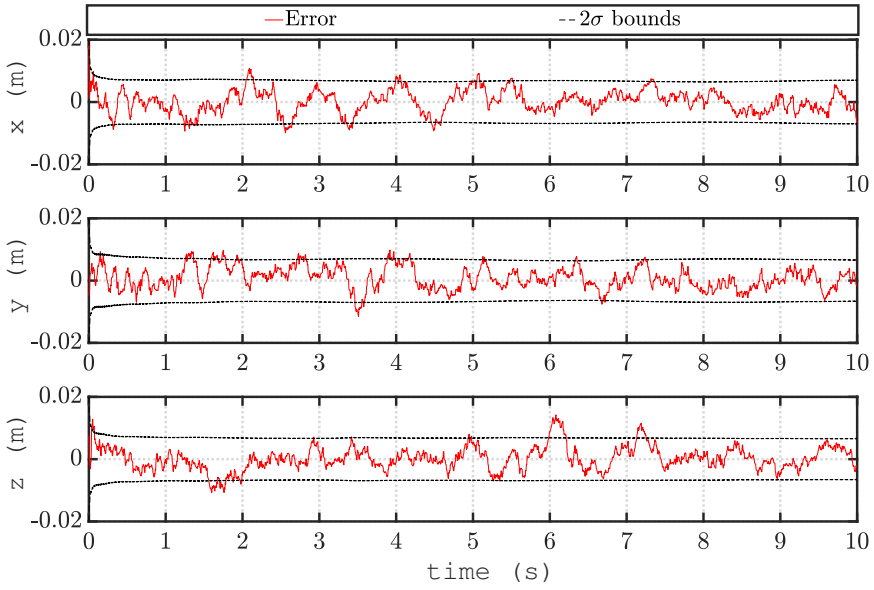


(a) Estimation error of angles using 1-D magnetometer measurements.

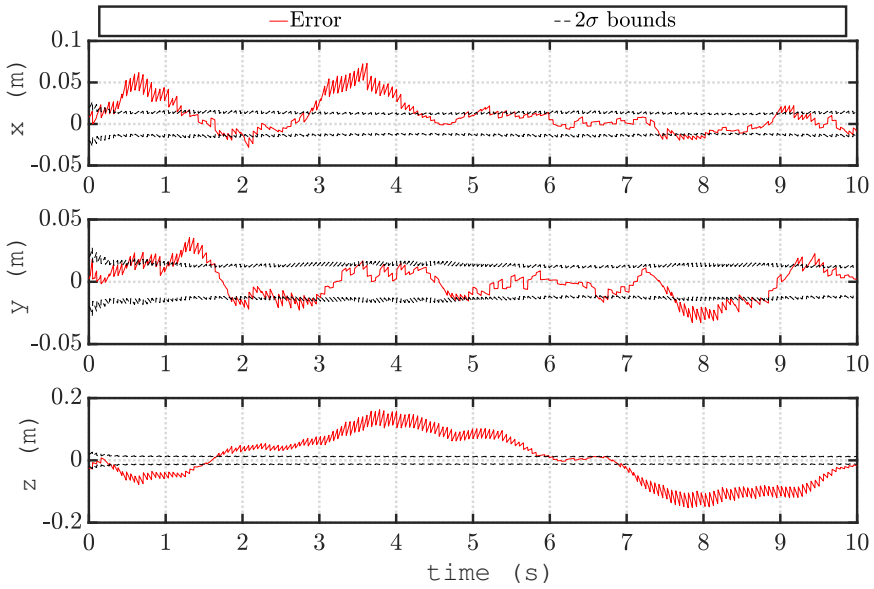


(b) Estimation error of angles using 3-D magnetometer measurements.

Figure D.2: Figures show the estimation error of vehicle angles using a magnetometer as 1-D and 3-D sensors under unknown local magnetic disturbance.



(a) Estimation error of positions using 1-D magnetometer measurements.



(b) Estimation error of positions using 3-D magnetometer measurements.

Figure D.3: Figures show the estimation error of vehicle angles using a magnetometer as 1-D and 3-D sensors under unknown local magnetic disturbance.

APPENDIX E

SEQUENTIAL UPDATE

The sequential Kalman filter is a way of implementing the Kalman filter without matrix inversion. When the measurement noise covariance matrix Σ is diagonal and constant, the measurement update can be done sequentially. Let a measurement z is a r-dimensional vector. Then, for $i = 1, \ldots, r$, (2.105), (2.106), and (2.107) are written as follows:

$$\hat{x}_{i,k}^{+} = \hat{x}_{i-1,k}^{+} + K_{i,k}(z_{i,k} - H_{i,k}\hat{x}_{i-1,k}^{T}), \tag{E.1}$$

$$P_{i,k}^{+} = P_{i-1,k}^{+} - K_{i,k}H_{i,k}P_{i-1,k}^{+}, \tag{E.2}$$

$$K_{i,k} = P_{i-1,k}^{+}H_{i,k}^{T}(H_{i,k}P_{i-1,k}^{+}H_{i,k}^{T} + \Sigma_{i,k})^{-1}, \tag{E.3}$$

where

$$\Sigma_k = \begin{bmatrix} \Sigma_{1,k} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \Sigma_{r,k} \end{bmatrix}. \tag{E.4}$$

In order for the algorithms to be simply written, the *a posteriori* states and covariance matrix with zero-th measurement are defined as follows:

$$\hat{x}_{0,k}^{+} \triangleq \hat{x}_{k}^{-}, \tag{E.5}$$

$$P_{0,k}^{+} \triangleq P_{k}^{-}. \tag{E.6}$$

Using this method, $H_{i,k}P^{+}_{i-1,k}H^{T}_{i,k} + \Sigma_{i,k}$ of (E.3) becomes a scalar; therefore,

$$\begin{aligned} K_{i,k} &= P^{+}_{i-1,k}H^{T}_{i,k}(H_{i,k}P^{+}_{i-1,k}H^{T}_{i,k} + \Sigma_{i,k})^{-1} \\ &= \frac{P^{+}_{i-1,k}H^{T}_{i,k}}{H_{i,k}P^{+}_{i-1,k}H^{T}_{i,k} + \Sigma_{i,k}}. \end{aligned} \tag{E.7}$$

This way, matrix inversion is not necessary. The sequential update method is implemented in C++ and MATLAB using GPS measurements (3.7) since GPS is a 6-D measurement and displays the most notable difference. Table E.1 summarizes the results of this experiment. In C++, matrix operations use a C++ library called *Eigen* [132], and MATLAB uses default functions including matrix inversion. In both C++ and MATLAB, the standard update method outperforms over the sequential method. Since the extended Kalman filter (EKF) requires the recalculation of the Jacobian matrix for each update (e.g. GPS updates need to calculate Jacobian six times), the sequential update does not always improve the computational cost. Considering the results of this experiment, the standard measurement update is used throughout this paper.

Table E.1: **Computational Cost for GPS Measurement Update**

	C++	MATLAB
Standard update	9.2 μs	150 μs
Sequential update	13.4 μs	400 μs

REFERENCES

[1] D. Moormann, "DHL parcelcopter research flight campaign 2014 for emergency delivery of medication," in *Proceedings of the ICAO RPAS Symposium*, 2015.

[2] C. Airplanes, "Statistical summary of commercial jet airplane accidents," *Worldwide Operations*, vol. 2008, 1959.

[3] T. Muskardin, G. Balmer, S. Wlach, K. Kondak, M. Laiacker, and A. Ollero, "Landing of a fixed-wing UAV on a mobile ground vehicle," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1237–1242.

[4] T. Merz, S. Duranti, and G. Conte, "Autonomous landing of an unmanned helicopter based on vision and inertial sensing," *Experimental Robotics IX*, pp. 343–352, 2006.

[5] S. Lange, N. Sunderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, IEEE, 2009, pp. 1–6.

[6] M. Verbandt, B. Theys, and J. De Schutter, "Robust marker-tracking system for vision-based autonomous landing of VTOL UAVs," 2014.

[7] Y. Jung, D. Lee, and H. Bang, "Close-range vision navigation and guidance for rotary UAV autonomous landing," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, IEEE, 2015, pp. 342–347.

[8] C. S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, IEEE, vol. 2, 2001, pp. 1720–1727.

[9] G. J. Zhang and F. Q. Zhou, "Position and orientation estimation method for landing of unmanned aerial vehicle with two circle based computer vision," *Acta Aeronautica Et Astronautica Sinica*, vol. 3, p. 017, 2005.

[10] J. Kim, Y. Jung, D. Lee, and D. H. Shim, "Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, IEEE, 2014, pp. 1243–1252.

[11] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Visually guided landing of an unmanned aerial vehicle," *IEEE transactions on robotics and automation*, vol. 19, no. 3, pp. 371–380, 2003.

[12] ——, "Vision-based autonomous landing of an unmanned aerial vehicle," in *Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on*, IEEE, vol. 3, 2002, pp. 2799–2804.

[13] Y. Fan, S. Haiqing, and W. Hong, "A vision-based algorithm for landing unmanned aerial vehicles," in *Computer Science and Software Engineering, 2008 International Conference on*, IEEE, vol. 1, 2008, pp. 993–996.

[14] S. Yang, S. A. Scherer, K. Schauwecker, and A. Zell, "Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, p. 27, 2014.

[15] A. C. Tsai, P. W. Gibbens, and R. H. Stone, "Terminal phase vision-based target recognition and 3D pose estimation for a tail-sitter, vertical takeoff and landing unmanned air vehicle," in *Pacific-Rim Symposium on Image and Video Technology*, Springer, 2006, pp. 672–681.

[16] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3400–3407.

[17] A. Borowczyk, D.-T. Nguyen, A. Phu-Van Nguyen, D. Q. Nguyen, D. Saussié, and J. Le Ny, "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 488–10 494, 2017.

[18] K. Ling, "Precision landing of a quadrotor UAV on a moving target using low-cost sensors," Master's thesis, University of Waterloo, 2014.

[19] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 971–976.

[20] P. Serra, R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre, "Landing on a moving target using image-based visual servo control," in *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 2179–2184.

[21] T. Nakamura, S. Haviland, D. Bershadsky, D. Magree, and E. N. Johnson, "Vision-based closed-loop tracking using micro air vehicles," in *Aerospace Conference, 2016 IEEE*, IEEE, 2016, pp. 1–12.

[22] T. Nakamura, S. Haviland, D. Bershadsky, and E. N. Johnson, "Vision-based optimal landing on a moving platform," in *72nd American Helicopter Society International Annual Forum*, West Palm Beach, Frorida, 2016.

[23] T. Nakamura and E. N. Johnson, "Vision-based multiple model adaptive estimation of ground targets from airborne images," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, IEEE, 2016, pp. 598–607.

[24] T. Nakamura, D. Magree, and E. N. Johnson, "Estimation techniques in robust vision-based landing of aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 664–11 669, 2017.

[25] T. Nakamura, S. Haviland, D. Bershadsky, and E. N. Johnson, "Vision sensor fusion for autonomous landing," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2017, p. 0674.

[26] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, IEEE, vol. 1, 2001, pp. I–511.

[27] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, IEEE, vol. 1, 2002, pp. I–I.

[28] Q. Ren, P. Li, and B. Han, "Vision-based mini unmanned helicopter pose and position estimation," *Journal of Zhejiang University (Engineering Science)*, vol. 43, no. 1, pp. 18–22, 2009.

[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, vol. 1, 2005, pp. 886–893.

[30] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[31] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[32] Y. Li, Y. R. Wang, H. Luo, Y. Chen, and Y. Y. Jiang, "Landmark recognition for UAV autonomous landing based on vision," *Jisuanji Yingyong Yanjiu*, vol. 29, no. 7, pp. 2780–2783, 2012.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[34] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *ArXiv preprint arXiv:1612.08242*, 2016.

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[36] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[38] C. Teuliere, L. Eck, and E. Marchand, "Chasing a moving target from a flying UAV," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 4929–4934.

[39] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on automatic control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[40] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, IEEE, 2008, pp. 340–345.

[41] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, IEEE, 2000, pp. 153–158.

[42] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052 –1067, 2007.

[43] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, 1403 –1410 vol.2.

[44] E. Eade and T. Drummond, "Scalable monocular SLAM," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, 2006, pp. 469 –476.

[45] S. Yang, J. Ying, Y. Lu, and Z. Li, "Precise quadrotor autonomous landing with SRUKF vision perception," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 2196–2201.

[46] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[47] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality, 2007. 6th IEEE and ACM International Symposium on*, 2007, pp. 225 –234.

[48] D. Magree and E. N. Johnson, "A monocular vision-aided inertial navigation system with improved numerical stability," in *Proceedings of the AIAA Guidance Navigation and Control Conference, Kissimmee, Florida*, 2015, pp. 6–2015.

[49] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held RGB-D camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, vol. 180, 2011, pp. 1–15.

[50] A. A. B. Pritsker and J. J. O'Reilly, *Simulation with visual SLAM and AweSim*. John Wiley & Sons, 1999.

[51] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference, Nottingham, September 1-5, 2014*, BMVA Press, 2014.

[52] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[53] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, IEEE, 2010, pp. 2544–2550.

[54] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1090–1097.

[55] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[56] K. P. Murphy, "Bayesian map learning in dynamic environments," in *Advances in Neural Information Processing Systems*, 2000, pp. 1015–1021.

[57] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 2000, pp. 176–183.

[58] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[59] H. Cho, P. E. Rybski, and W. Zhang, "Vision-based 3D bicycle tracking using deformable part model and interacting multiple model filter," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 4391–4398.

[60] R. M. Murray, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[61] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[62] F. S. Grassia, "Practical parameterization of rotations using the exponential map," *Journal of graphics tools*, vol. 3, no. 3, pp. 29–48, 1998.

[63] G. Gallego and A. Yezzi, "A compact formula for the derivative of a 3-D rotation in exponential coordinates," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 3, pp. 378–384, 2015.

[64] O. A. Bauchau and L. Trainelli, "The vectorial parameterization of rotation," *Nonlinear dynamics*, vol. 32, no. 1, pp. 71–92, 2003.

[65] A. D. Wu, E. N. Johnson, M. Kaess, F. Dellart, and G. Chowdhary, "Autonomous flight in GPS-denied environments using monocular vision and inertial sensors," *Journal of Aerospace Information Systems*, vol. 10, no. 4, pp. 172–186, 2013.

[66] M Ritto-Corrêa and D Camotim, "On the differentiation of the Rodrigues formula and its significance for the vector-like parameterization of reissner–simo beam theory," *International Journal for Numerical Methods in Engineering*, vol. 55, no. 9, pp. 1005–1032, 2002.

[67] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-D vision: From images to geometric models*. Springer Science & Business Media, 2012, vol. 26.

[68] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer vision in C++ with the OpenCV library*. " O'Reilly Media, Inc.", 2016.

[69] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles," *IEEE Robotics and Automation magazine*, vol. 20, no. 32, 2012.

[70] R. M. Brannon, "Rotation, a review of useful theorems involving proper orthogonal matrices referenced to three dimensional physical space," *Sandia National Laboratories, Albuquerque, NM*, 2002.

[71] J. Farrell, *Aided navigation: GPS with high rate sensors*. McGraw-Hill, Inc., 2008.

[72] R. M. Rogers, *Applied mathematics in integrated navigation systems*. AIAA, 2003, vol. 1.

[73] B. Hall, *Lie groups, Lie algebras, and representations: An elementary introduction*. Springer, 2015, vol. 222.

[74] S. Helgason, *Differential geometry and symmetric spaces*. American Mathematical Soc., 2001, vol. 341.

[75] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011, vol. 2.

[76] E. Eade, "Lie groups for 2D and 3D transformations," *URL http://ethaneade. com/lie. pdf, revised Dec*, 2013.

[77] ——, *Lie groups for computer vision*, 2014.

[78] P Stein, "Some general theorems on iterants," *J. Res. Nat. Bur. Standards*, vol. 48, no. 1, pp. 82–83, 1952.

[79] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[80] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[81] A. I. Dale, *A history of inverse probability: From Thomas Bayes to Karl Pearson*. Springer Science & Business Media, 2012.

[82] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-time visual-inertial odometry with event cameras," *IEEE Trans. Robot*, 2017.

[83] A. Patron-Perez, S. Lovegrove, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," *International Journal of Computer Vision*, vol. 113, no. 3, pp. 208–219, 2015.

[84] A. Martinelli, "Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 44–60, 2012.

[85] E. J. Ohlmeyer, "Analysis of an ultra-tightly coupled GPS/INS system in jamming," in *Position, Location, And Navigation Symposium, 2006 IEEE/ION*, IEEE, 2006, pp. 44–53.

[86] E. Eade, "Monocular simultaneous localisation and mapping," PhD thesis, University of Cambridge, 2009.

[87] H. D. Black, "A passive system for determining the attitude of a satellite," *AIAA Journal*, vol. 2, no. 7, pp. 1350–1351, 1964.

[88] ——, "Early development of transit, the navy navigation satellite system," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 4, pp. 577–585, 1990.

[89] C. D. Hall, "Spacecraft attitude dynamics and control," *Lecture Notes posted on Handouts page [online]*, vol. 12, no. 2003, 2003.

[90] F. L. Markley, "Attitude determination using vector observations and the singular value decomposition," *Journal of the Astronautical Sciences*, vol. 36, no. 3, pp. 245–258, 1988.

[91] M. Figueirôa, A. Moutinho, and J. R. Azinheira, "Attitude estimation in SO (3): A comparative case study," in *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*, IEEE, 2014, pp. 79–84.

[92] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings F (Radar and Signal Processing)*, IET, vol. 140, 1993, pp. 107–113.

[93] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of computational and graphical statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[94] J. S. Liu, *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[95] B. Ristic, S. Arulampalam, and N. J. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2004.

[96] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-Blackwellized particle filter for multiple target tracking," *Information Fusion*, vol. 8, no. 1, pp. 2–15, 2007.

[97] F. Daum and J. Huang, "Curse of dimensionality and particle filters," in *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, IEEE, vol. 4, 2003, 4_1979–4_1993.

[98] C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson, "Obstacles to high-dimensional particle filtering," *Monthly Weather Review*, vol. 136, no. 12, pp. 4629–4640, 2008.

[99] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[100] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. A. Wan, "The unscented particle filter," in *Advances in neural information processing systems*, 2001, pp. 584–590.

[101] T. Nakamura, S. Haviland, D. Bershadsky, D. Magree, and E. N. Johnson, "Particle filter for closed-loop detection and tracking from occluded airborne images," in *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*, IEEE, 2017, pp. 256–263.

[102] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[103] J. S. Liu and R. Chen, "Blind deconvolution via sequential imputations," *Journal of the american statistical association*, vol. 90, no. 430, pp. 567–576, 1995.

[104] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2013, vol. 3.

[105] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International Journal of Computer Vision*, vol. 103, no. 3, pp. 267–305, 2013.

[106] J. H. Manton, "A globally convergent numerical algorithm for computing the centre of mass on compact Lie groups," in *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, IEEE, vol. 3, 2004, pp. 2211–2216.

[107] J. S. Liu, W. H. Wong, and A. Kong, "Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes," *Biometrika*, vol. 81, no. 1, pp. 27–40, 1994.

[108] A. E. Gelfand and A. F. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American Statistical Association*, vol. 85, no. 410, pp. 398–409, 1990.

[109] G. Casella and C. P. Robert, "Rao-Blackwellisation of sampling schemes," *Biometrika*, vol. 83, no. 1, pp. 81–94, 1996.

[110] E. L. Lehmann and G. Casella, *Theory of point estimation*. Springer Science & Business Media, 2006.

[111] J. O. Berger, *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[112] H. Akashi and H. Kumamoto, "Random sampling approach to state estimation in switching environments," *Automatica*, vol. 13, no. 4, pp. 429–434, 1977.

[113] S.-M. Oh, *Nonlinear estimation for vision-based air-to-air tracking*. Georgia Institute of Technology, 2007.

[114] R. Chen and J. S. Liu, "Mixture Kalman filters," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 62, no. 3, pp. 493–508, 2000.

[115] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The secrets of salient object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 280–287.

[116] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, IEEE, vol. 2, 2006, pp. 1964–1971.

[117] H. Carr, J. Snoeyink, and U. Axen, "Computing contour trees in all dimensions," *Computational Geometry*, vol. 24, no. 2, pp. 75–94, 2003.

[118] J. P. Lewis, "Fast template matching," in *Vision interface*, vol. 95, 1995, pp. 15–19.

[119] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.," *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[120] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[121] E. N. Johnson, J. G. Mooney, and H. B. Christophersen, "Fourteen years of autonomous rotorcraft research at the Georgia Institute of Technology," in *Proceedings of the 2nd Asian/Australian Rotorcraft Forum and the 4th International Basic Research Conference on Rotorcraft Technology*, 2013.

[122] E. N. Johnson and S. K. Kannan, "Adaptive trajectory control for autonomous helicopters," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 3, pp. 524–538, May 2005.

[123] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ros and gazebo," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Springer, 2012, pp. 400–411.

[124] E. N. Johnson and D. P. Schrage, "System integration and operation of a research unmanned aerial vehicle," *AIAA Journal of Aerospace Computing, Information and Communication*, vol. 1, no. 1, pp. 5–18, 2004.

[125] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-Blackwellized particle filter for multiple target tracking," *Information Fusion*, vol. 8, no. 1, pp. 2–15, 2007.

[126] J. Hartikainen and S. Särkkä, "RBMCDAbox-Matlab toolbox of Rao-Blackwellized data association particle filters," *Documentation of RBMCDA Toolbox for Matlab V*, 2008.

[127] L. D. Stone, T. L. Corwin, and C. A. Barlow, "Bayesian multiple target tracking, Artech House," *Inc., Norwood, MA*, vol. 2062, 1999.

[128] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DOF camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[129] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[130] B. Etkin, *Dynamics of atmospheric flight*. Courier Corporation, 2012.

[131] T. Nakamura and E. N. Johnson, "Trade studies on implementation of extended Kalman filters for sUAS navigation," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, American Institute of Aeronautics and Astronautics, 2019.

[132] G. Guennebaud, B. Jacob, *et al.*, *Eigen v3*, http://eigen.tuxfamily.org, 2010.

VITA

Takuma Nakamura received his Ph.D. from the Georgia Institute of Technology (2018) after working with Dr. Eric N. Johnson in the Unmanned Aerial Vehicle Research Facility at Georgia Tech. A graduate of Tohoku University in Sendai, Japan (BE Mechanical and Aerospace Engineering, 2013), he also holds a Master of Science in Aerospace Engineering from Georgia Tech (2015). An FAA licensed private and sUAS pilot, Nakamura's research interests include sensor fusion, computer vision systems, autonomous navigation for UAVs, and flight simulation. He will join Amazon Prime Air as a research scientist to achieve an autonomous package delivery using UAVs. His experience as a research intern at Amazon Prime Air and as a human-powered airplane pilot contribute to his expertise.